ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO INDUSTRIAL

Título del proyecto:

ANALIZADOR DE ESPECTRO SOBRE FPGA

MEMORIA

Bardera Barbolla, David

Corres Sanz, Jesús María

Pamplona, junio del 2012

Agradecimientos

A ti madre, por recibirme cada mañana con una sonrisa, por confiar y creer siempre en mi, por protegerme, por quererme como nadie, por apoyarme incondicionalmente anteponiendo mis interés, por todas esas pequeñas y grandes cosas que te hacen única, gracias por siempre, te quiero.

A ti padre por estar siempre y saber que cuento contigo para todo; por trabajar por nosotros para que seamos mejor personas y por inculcarme desde pequeño esa semilla del querer saber.

A ti hermano porque por mucho que nos peleemos nunca nos separaremos.

A Alberto por los días de estudio contigo, porque casi se convertían en un "placer".

A Alberto y a Cristian por enseñarme siempre cosas nuevas, diferentes, por nuestras conversaciones, siempre os llevare conmigo.

Nota especial para Alejandro, por los buenos ratos pasados en todos estos años contigo, porque con tu voz alegrabas las noches en el piso, por estar siempre dispuesto a pegarte.

A Carlos por el equipo de estudio cuando vivíamos en la misma calle.

A ti Hannah por cuidar tan bien de mí y de Juanito, por amarme.

A Móises por los momentos pasados allá en Pinar; gracias por todo.

A Bárbaro por toda la ayuda recibida durante la ejecución del proyecto y a toda su familia por tratarme como uno más de ellos cuando estaba lejos de casa.

A Susana y el resto de personas del Departamento de Cooperación al Desarrollo por poder haber hecho posible mi participación en este proyecto.

Resumen

La ejecución de este Proyecto Fin de Carrera parte de una beca de Cooperación al Desarrollo del Gobierno de Navarra, consistente en una estancia en la Universidad Hermanos Saíz Montes de Oca de Pinar del Río (Cuba).

Esta memoria aborda el diseño e implantación de un analizador de espectro a bajo coste, con el objeto de poder ser utilizado en centros educativos y profesionales de países con recursos económicos limitados. De esta manera, se propone realizar un analizador digital de espectro embebido en FPGA para el estudio de señales audio.

En el documento se presenta primeramente un estudio de los analizadores comerciales y de los sistemas digitales disponibles en el mercado; así como una descripción de las características de los FPGAs. A continuación se detalla la fundamentación teórica de las herramientas usadas en el proceso de diseño e implantación y los módulos de diseño analógico y programación de software realizado en VHDL.

Como resultado se tiene un analizador de espectro digital sobre FPGA capaz de medir el espectro completo de audio y ofrecer la posibilidad de elegir diferentes opciones de span, enventanado y escalado de ejes.

Índice

Capítulo 1

Introducción

1.1 Contextualización

La realización de este proyecto fin de carrera parte de una beca de Cooperación al Desarrollo de la Universidad Pública de Navarra; por la cual, gran parte del trabajo se desarrolla en la Universidad Hermanos Saíz Montes de Oca de Pinar del Río (Cuba). Atendiendo a estas circunstancias se hace la siguiente justificación de la temática sobre la que versa el proyecto.

Los diseñadores expertos de hardware, las empresas que se dedican al desarrollo de tecnologías en el campo de la electrónica y las instituciones educativas que imparten carreras técnicas e ingenierías que tengan relación con la electrónica o el diseño de sistemas, necesitan utilizar analizadores de espectros en el análisis de las señales. Algunas instituciones educativas no cuentan con el equipamiento necesario en sus laboratorios para que el alumnado realice tanto prácticas de laboratorio como proyectos curriculares y extracurriculares de una manera adecuada. Esto ocurre, principalmente, por el alto coste en el mercado mundial de los diferentes instrumentos electrónicos de medición (osciloscopios, multímetros, analizadores lógicos, generadores de funciones, analizadores de espectros, etc.), por lo que sólo es posible adquirir un número reducido de los que se deberían necesitar. Además, en la mayoría de los casos, las prácticas o diseños que realizan los alumnos no explotan al máximo todas las características con las que cuenta el equipo de medición. Es por ello que se plantea la necesidad de equipar a las instituciones educativas, centros de investigación, industrias, etc. del país, y específicamente de la provincia de Pinar del Río, con instrumentos de medición electrónica, fundamentalmente analizadores de espectro digitales, que se ajusten a sus necesidades de diseño y a un coste mucho más bajo que los equipos comerciales antes mencionados.

El desarrollo de esta investigación parte de la hipótesis de que una vez que se logre el diseño del analizador de espectro para el análisis en frecuencia de señales sobre un FPGA, teniendo en cuenta la frecuencia de éstas, las características de los Dispositivos Lógicos Programables FPGA, el Lenguaje de Descripción de Hardware utilizado, el medio que visualizará los resultados de las mismas y el tiempo, entonces, se podrá implementar para su utilización en investigaciones y aplicaciones en centros educativos y profesionales con perfiles electrónicos, los cuales podrán contar con estos ejemplares de bajo coste, propiciándose el desarrollo científico-tecnológico de los mismos.

Por otra parte, a título personal la elaboración de este proyecto fin de carrera ha supuesto una iniciación en la electrónica digital, rama de la ingeniería y de la tecnología que nos rodea de la cual había adquirido escasos conocimientos a lo largo de los estudios de Ingeniería Industrial. De esta manera, he profundizado en el estudio de aspectos

relacionados con el procesamiento de señales, la programación en VHDL, representación en VGA y la electrónica digital.

1.2 Estructura de la memoria

La memoria del proyecto fin de carrera se estructura en una serie de capítulos según su temática. De esta manera, se comienza con una introducción al proyecto que se ha realizado, explicando cuales son los objetivos que se pretenden, así como la situación que los condiciona.

En el Capítulo 2 se realiza una revisión teórica de los aspectos fundamentales sobre los que se ha investigado para adquirir los conocimientos imprescindibles en el desarrollo e implantación del analizador de espectro. Conforme a un orden preestablecido, se parte definiendo qué es un analizador de espectro y para qué se utiliza. Posteriormente, se explica qué son las señales y en qué consiste el procesamiento digital de ellas. Este punto lleva al siguiente en el que se introduce los sistemas digitales y se profundiza en el conocimiento de los FPGAs. Por último se muestran los fundamentos teóricos de la transformada discreta de Fourier; en particular del algoritmo Radix-2, y de la norma de video SVGA que se utiliza para la representación de los resultados en un monitor externo. Para una mejor comprensión de los aspectos teóricos, se ha decidido que conforme se avance en el desarrollo de las explicaciones y justificaciones de cada una de las partes físicas de que consta el proyecto, se detallará la fundamentación teórica relacionada.

El Capítulo 3 describe los diseños iniciales y simulaciones que han contribuido a definir las características del analizador de espectro. En el proceso de descripción se describen los fundamentos teóricos de las diferentes posibilidades válidas planteadas; así como las justificaciones sobre la opción elegida.

El Capitulo 4 se centra en el diseño e implantación final del sistema desarrollado. Partiendo de la estructura general se profundiza en cada una de las partes explicando su objetivo y funcionamiento. Asimismo se introducen los cálculos realizados en el diseño, esquemas para mejorar la explicación, partes más representativas del código de programación y esquemas de las conexiones realizadas.

El Capitulo 5 guarda una colección de los resultados obtenidos y de las pruebas realizadas para validar el funcionamiento del sistemas.

El Capítulo 6 trata el análisis económico del proyecto por medio de un presupuesto de los costes asociados; justificando el cumplimiento del objetivo de bajo coste.

Finalmente se muestran las Conclusiones Generales, Recomendaciones y Trabajos Futuros, Bibliografía y Anexo.

1.3 Objetivo General

El objetivo principal de este proyecto fin de carrera es el diseño e implantación de un analizador digital de espectro de bajo coste para el análisis espectral de señales eléctricas periódicas de audio comprendidas en el rango de (0,20)kHz y con una tensión de entrada en el rango de (5,-5)V y su posterior visualización en un monitor externo.

La programación se realiza en código VHDL y su implantación en una arquitectura FPGA embebida en un Kit que permite el uso de diversos periféricos, como es el puerto VGA que nos facilita la visualización en un monitor.

1.4 Objetivos Específicos

- Fundamentar teóricamente la investigación.
- Diseñar el interfaz de acondicionamiento de la señal conforme a las características del conversor A/D para la apropiada digitalización de la señal analógica.
- Diseñar los algoritmos para adquisición de señales, procesamiento y visualización a través de un monitor VGA.
- Diseñar los algoritmos necesarios para la implantación de un sistema que permita extraer las características espectrales de una señal periódica variante en el tiempo por medio del algoritmo de la FFT.
- Diseñar y simular los módulos VHDL que permitan aumentar las prestaciones del analizador de espectro diseñado, por ejemplo: distintas opciones de enventanado, zoom espectral, etc.
- Comprobar la funcionalidad de los diseños hardware mediante simulaciones en el ordenador a través de diversos software y mediante otros equipos como osciloscopios y analizadores de espectros de ámbito comercial.
- Validar el sistema mediante su funcionamiento práctico-experimental.

Por último, cabe añadir que la arquitectura que responde a dicho sistema debe ser flexible y de fácil adaptación para la realización de medidas en distintos rangos de frecuencia, pudiendo servir para el análisis de señales no solo de audio, sino de diferentes disciplinas como puede ser la mecánica en el análisis de vibraciones, la medicina, la sismología, etc. Asimismo debe permitir versatilidad a la hora de disponer de un FPGA determinado.

Capítulo 2

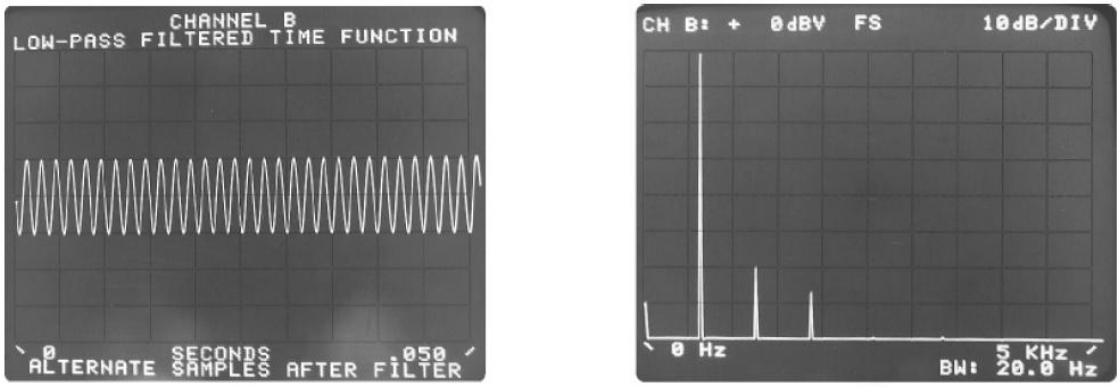
Aspectos teóricos previos

2.1 ANALIZADOR DE ESPECTRO

2.1.1 Introducción al analizador de espectro

Las señales eléctricas presentes en nuestro medio nos pueden brindar y transmitir diferentes tipos de información, que se podrá adquirir o entender dependiendo del análisis que se les haga. Es así, que en el dominio temporal se puede obtener información de una señal eléctrica tal como su amplitud, período y fase, siendo éstas las principales características de la misma.

El análisis de señales en el campo transformado permite descubrir aspectos de la señal que serían muy difíciles o imposibles de observar a partir de su representación temporal. Por ejemplo la Figura 2.1 (a) muestra un tono que parece ser sinusoidal puro. Sin embargo, el espectro de esta señal, que se observa en la Figura 2.1 (b), revela la presencia de otras componentes frecuenciales. Cuando estas componentes se grafican en el dominio frecuencial son fáciles de detectar porque no quedan enmascaradas por las señales de gran amplitud. Esto no significa que las mediciones en el dominio transformado sean "mejores" que las mediciones en el dominio tiempo. Cierta clase de medidas, como el tiempo de crecida y de caída de un pulso, el sobrepico, y las oscilaciones amortiguadas (*ringing*) de una señal sólo pueden medirse en el dominio temporal (por ejemplo, con un osciloscopio).



(a) (b)

Figura 2.1 **a)** Muestra tomada de un osciloscopio
b) Misma muestra tomada de un analizador de espectros

En otras aplicaciones, en cambio, las mediciones en el dominio frecuencial son más ventajosas, como por ejemplo la determinación del contenido armónico. En el campo de las comunicaciones es muy importante la medición de la distorsión armónica; tal es el caso de los sistemas celulares, donde debe verificarse el contenido armónico de la señal

portadora para evitar interferencias con otros sistemas que operan en las mismas frecuencias que los armónicos.

Mientras que el comportamiento de una señal en el dominio tiempo puede analizarse con un osciloscopio, el análisis en el dominio transformado requiere de un instrumento que tradicionalmente era bastante más costoso y más complicado de utilizar: el analizador de espectro. Hoy en día es frecuente que los osciloscopios digitales sean capaces de calcular el espectro de una señal, utilizando herramientas que posteriormente se trataran. Tales instrumentos reciben el nombre de analizadores digitales de espectro. A continuación se detallan los tipos más comunes que se pueden encontrar.

2.1.2 El analizador de espectro y los diferentes tipos disponibles

Un analizador de espectro es un equipo de medición electrónica que permite visualizar en una pantalla las componentes espectrales de una determinada señal presente en la entrada, pudiendo ser ésta cualquier tipo de onda eléctrica, acústica u óptica.

El eje de ordenadas suele presentarse en una escala logarítmica en dB el nivel del contenido espectral de la señal. En el eje de abscisas se representa la frecuencia, en una escala que es función de la separación temporal y del número de muestras capturadas. Se denomina frecuencia central del analizador a la que corresponde con la frecuencia en el punto medio de la pantalla.

Atendiendo a su naturaleza, los analizadores de espectro se pueden clasificar en dos tipos: analógicos y digitales:

- **Los analizadores espectrales analógicos** se dividen en dos clases. La primera clase tiene varias salidas, y en un tiempo t = t0 cada una de ellas mide una componente frecuencial distinta de la señal de entrada; de modo que el espectro deseado se lee simultáneamente. Se distinguen según su construcción:

  Analizador de filtros en paralelo. Se trata de un analizador en tiempo real formado por un banco de filtros paso banda contiguos y un banco de detectores, donde cada conjunto analiza una porción del espectro.

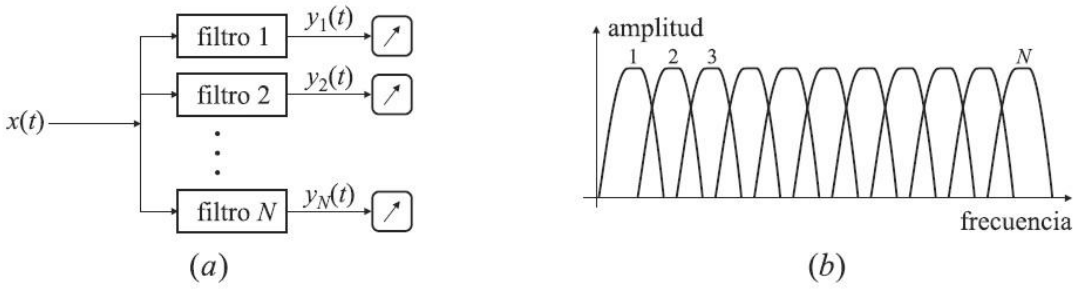


Figura 2.2 Analizador analógico de espectro simultáneo

La segunda clase consta de una única salida que mide las distintas componentes frecuenciales en instantes de tiempos consecutivos. Se dice que el espectro se determina secuencialmente. Destacan:

Analizador secuencial. Aprovecha el mismo principio que el primero, pero emplea un conmutador para reducir el número de detectores necesarios.

Analizador de barrido con filtro sintonizable. En este sistema (Figura 2.3) el analizador dispone de un filtro (de ancho de banda B) que se sintoniza linealmente a lo largo de un margen frecuencial (f1-f3) durante un intervalo de tiempo T, manteniendo durante este intervalo su ancho de banda constante.

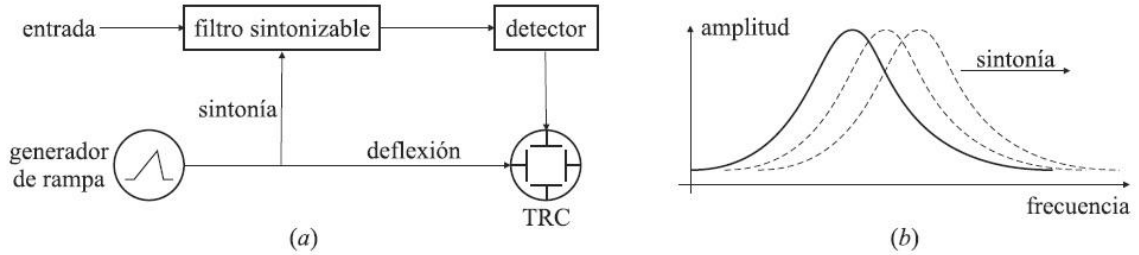


Figura 2.3 Analizador analógico de espectro de barrido

Analizador superheterodino. Si en lugar de sintonizar la frecuencia central del filtro como en el caso del analizador de barrido con filtro sintonizable, lo que se hace es desplazar el espectro de la señal de entrada, el resultado que se conseguiría sería el mismo. Basándose en esta idea, el analizador superheterodino realiza una doble conversión a frecuencia intermedia (FI), permitiendo utilizar un único filtro, de manera que éste mantiene sus características a lo largo de toda la banda de frecuencias. No obstante permite disponer de diferentes filtros seleccionables en la etapa de FI para poder trabajar con diferentes resoluciones, según el tipo de medida que se desee realizar.

Algunos otros analizadores como los de *Tektronix* utilizan un híbrido entre análogo y digital al que llaman analizador de espectro en "tiempo real".

- **Un analizador digital de espectro** utiliza la FFT (*Fast Fourier Transform*), un proceso matemático que transforma una señal en sus componentes espectrales. Algunas medidas requieren que se preserve la información completa de señal: frecuencia y fase; este tipo de análisis se llama vectorial. Equipos como los de *Agilent Technologies* (antiguamente conocidos como *Hewlett Packard*) usan este tipo de análisis.

Por último, hacer un inciso para aclarar que estos equipos son sensibles a la señal de entrada y dan unas medidas muy precisas dentro de las especificaciones para las cuales fueron construidas. Sin embargo, la sensibilidad de la medida hace que los resultados pueden estar deteriorados respecto a la señal que se requiere analizar por distorsión o mal empleo al medir la señal de entrada. En algunos casos se debe tomar ciertas medidas de protección antes de analizar señales desconocidas.

2.2 PROCESAMIENTO DE SEÑALES

2.2.1 Introducción al procesamiento de señales

El procesamiento de señales es un área de la ciencia y la ingeniería que se ha desarrollado rápidamente durante los últimos 30 años. Este rápido desarrollo es el resultado de los avances tecnológicos tanto de los ordenadores digitales como en la fabricación de circuitos integrados.

Estos circuitos digitales económicos y relativamente rápidos, han hecho posible construir sistemas digitales altamente sofisticados; de aquí que muchas de las tareas de procesamiento de señal que convencionalmente se realizaban analógicamente se realicen hoy mediante hardware digital, más barato y fiable.

No obstante el procesamiento digital de señales no es siempre la solución apropiada; así, para muchas señales de gran ancho de banda se requiere procesamiento en tiempo real, siendo el procesamiento analógico e incluso óptico la única solución válida. Sin embargo, cuando los circuitos digitales se encuentran disponibles y tienen la velocidad suficiente, son preferibles.

El hardware digital y el software asociado proporcionan un mayor grado de flexibilidad en el diseño de sistemas, permitiendo operaciones programables.

2.2.2 Señales y sistemas

Una señal se define como una cantidad física que varía con el tiempo, el espacio o cualquier otra variable o variables independientes. Matemáticamente describimos una señal como una función de una o más variables independientes.

Figura 2.4 Señal de audio

Un sistema se puede definir como un dispositivo físico que realiza una operación sobre una señal. Por ejemplo, se denomina sistema a un filtro; el cual se utiliza para seleccionar la información deseada de una señal y reducir el ruido y las interferencias. Cuando se pasa una señal por un sistema decimos que hemos procesado la señal [Sys.1].

Para nuestros fines es conveniente ampliar la definición de sistema para incluir no sólo los dispositivos físicos, sino también realizaciones software de operaciones sobre una señal. Por ejemplo, un sistema mediante hardware digital (circuitos lógicos) configurado para

ejecutar las operaciones deseadas. De tal forma que se tiene un procesador digital mediante un sistema combinación de hardware digital y software.

Las operaciones realizadas por tales sistemas se especifican generalmente matemáticamente. El método o conjunto de reglas para implementar el sistema mediante un programa que ejecuta las operaciones matemáticas correspondientes se denomina algoritmo.

2.2.3 Elementos básicos de un sistema de procesamiento digital de señal

Definidos los términos de señal y sistema de procesamiento de señal, el análisis se centra en los sistemas de procesamiento de señales; más en concreto en los sistemas digitales DPS (*Digital Signal Processor*). En general, un sistema de estas características necesita interactuar con el exterior para recoger las señales analógicas que queremos procesar y posteriormente devolver estas señales a dominio analógico. Las etapas básicas de las que consta son (Figura 2.5):

- Conversión de la señal continua en tiempo y amplitud en una señal digital a través de un interfaz denominado conversor analógico-digital (A/D)
- Procesamiento de la señal
- Conversión de la señal digital procesada, en una señal continua

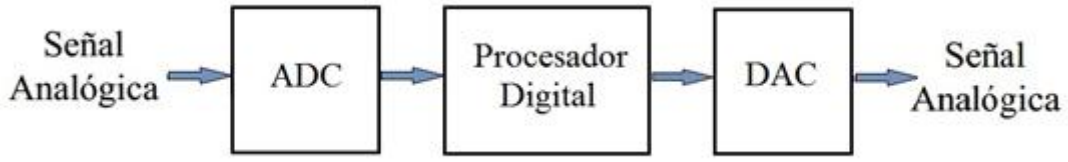


Figura 2.5 Esquema básico de un sistema de procesado de señal

El interfaz que posibilita la conversión desde el dominio digital al analógico se denomina, conversor digital-analógico (D/A). Un caso práctico es el utilizado en las señales de video transmitidas mediante la norma de video VGA, donde las señales se elaboran en formato digital desde el procesador y son enviadas a un monitor externo analógico.

En tal caso, se requiere de dicha conversión. Cabe señalar que existen tareas de procesamiento como las simulaciones o las síntesis de señales en las que no necesariamente estarán todas estas etapas.

2.2.4 Ventajas del procesamiento digital frente al analógico

En general las magnitudes medidas por los sistemas electrónicos son analógicas por naturaleza (presión, temperatura, humedad, etc). Por medio de sensores se obtienes señales eléctricas analógicas equivalentes. Es aquí donde los convertidores analógicos- digitales y los procesadores digitales juegan un papel fundamental, ya que trabajar con la información

en formato digital ofrece grandes ventajas sobre el formato analógico, entre la que se pueden destacar:

1. Programabilidad y flexibilidad. Un sistema digital programable facilita el cambio de los algoritmos sin necesidad de modificar el hardware como ocurre con los sistemas analógicos.

2. Repetitividad. La memoria y la lógica de un procesador no se alteran. Los procesos no son influenciados por derivas térmicas o tolerancias de los componentes.

3. Coste. En muchos casos la implementación digital del sistema es más barata que su equivalente analógico debido al hardware digital y a la flexibilidad ante las modificaciones que permite. Además, el cambio de funcionalidad o la corrección de errores puede llevarse a cabo mediante software, reduciendo drásticamente los costes

4. Almacenamiento. Las señales digitales se almacenan fácilmente en soporte magnético sin deterioro o pérdida en la fidelidad de la señal, aparte de la introducida en la conversión A/D.

5. El método digital de señales posibilita la implementación de algoritmos de procesado de señal sofisticados. Existen sistemas digitales sin equivalente analógico como los filtros FIR. Digitalmente se pueden generar formas de onda arbitrarias

6. Los dispositivos digitales tienen gran fiabilidad y precisión, mayor esperanza de vida y son más robustos ante las condiciones ambientales.

Por estos motivos la tecnología digital ha ido reemplazando a la analógica en muy diversos campos tales como televisión, fotografía o comunicaciones. En muchos de ellos, como ocurre en los sistemas de audio o de medida, la calidad de la señal es de vital importancia. Es en estos casos donde se debe prestar un especial interés al diseño de la etapa de conversión, ya que las transformaciones A/D o D/A son procesos que deterioran en gran medida la calidad de las señales.

Sin embargo, la implementación digital también tiene sus limitaciones; en concreto la velocidad de operación de los conversores A/D y procesadores.

2.2.5 Señales continuas frente a señales discretas

Existen dos tipos básicos de señales dependiendo de la naturaleza de la variable independiente (tiempo) que consideran: señales de tiempo continuo y discreto (Figura 2.6).

Si una señal toma todos los valores posible en un intervalo tanto finito como infinito, se dice que es continua. Si por el contrario toma valores de un conjunto finito de valores se

dice que es discreta. Normalmente estos valores son equidistantes y pueden expresarse como un múltiplo de la distancia entre dos valores sucesivos.

Una señal en tiempo discreto, que toma los valores en un conjunto discreto se denomina señal digital. Para que una señal analógica pueda ser procesada digitalmente ha de ser convertida a digital; es decir, ha de ser transformada en una secuencia de números de precisión finita. Este proceso se denomina conversión analógico-digital.

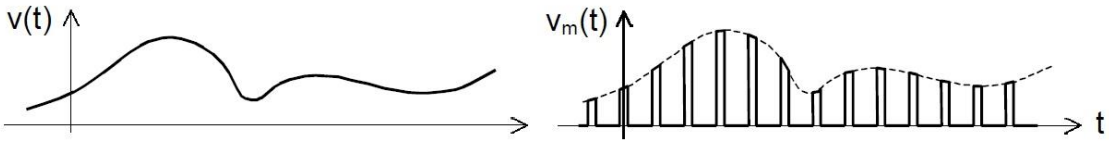


Figura 2.6 Señal continua frente a señal discreta

2.2.6 Conversión analógico-digital

La gran mayoría de las señales de interés práctico como señales de voz, señales biológicas, sísmicas, de radar y de diversos tipos de comunicación como audio, video, etc. son analógicas. En esta sección se describirá el proceso de conversión de señales analógicas a digitales. En la Figura 2.7 se muestra en un diagrama de bloques los tres pasos que se desarrollan en un convertidor (A/D), los cuales son:

1. **Muestreo:** Conversión de la señal de tiempo continuo $x_a(t)$ a una señal de tiempo discreto $x(n)$ obtenida tomando "muestras" de la señal en tiempo continuo en instantes discretos, de manera que $x(n)=x_a(nT)$, donde T se denomina el *intervalo de muestreo*.
2. **Cuantificación:** Conversión de una sucesión de muestras de amplitud continua en una sucesión de valores discretos preestablecidos según el código utilizado. Durante el proceso de cuantificación se mide el nivel de tensión de cada una de las muestras, obtenidas en el proceso de muestreo, y se les atribuye un valor finito (discreto) de amplitud, seleccionado por aproximación dentro de un margen de niveles previamente fijado. De esta manera la señal de tiempo discreto $x(n)$ se convierte en una señal de tiempo discreto con valores discretos $x_q(n)$ (señal digital). La diferencia entre estas dos señales se denomina *error de cuantificación*.
3. **Codificación:** Es la representación de cada valor de la señal digital $x_q(n)$ mediante un código binario de *b* bits.

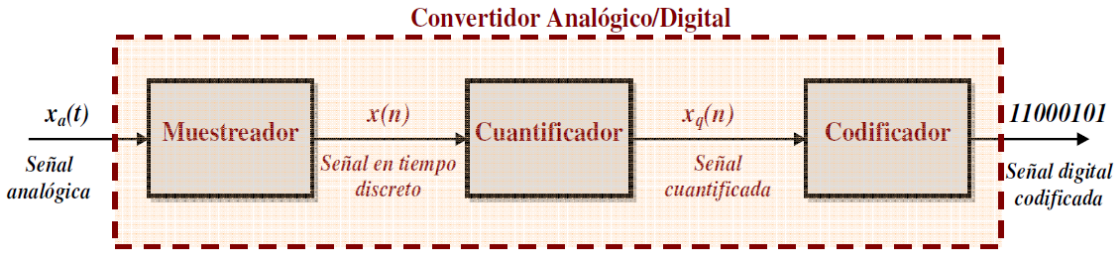


Figura 2.7 Esquema básico de un convertidor analógico-digital (A/D)

Al proceso de conversión de una señal digital a una analógica se le denomina conversión D/A o “reconstrucción de señales” y su función básica es “conectar” los puntos que representan cada valor de la señal digital efectuando algún tipo de interpolación.

2.2.7 Muestreo de señales analógicas

El proceso a través del cual una señal continua $x_a(t)$ es transformada en una señal discreta "equivalente" $x(n)$ consiste simplemente en la toma de muestras de la señal continua en instantes discretos de tiempo n denominados instantes de muestreo $n = \{...,-1,0,1,2,3,...\}$.

El tipo de muestreo que se considera por mayor uso y simplicidad se denomina muestreo periódico o uniforme y se muestra en la Figura 2.8. Para realizar dicho proceso es necesaria una señal de “reloj” que marque el ritmo de la toma de muestras; la frecuencia Fs de dicha señal se denomina frecuencia de muestreo (Hz) y su periodo $T=1/Fs$ se denomina periodo de muestreo y se considera constante.

Un sistema muestreador consiste simplemente en un switch que se cierra en el momento marcado por la señal de reloj y en todos los demás instantes permanece abierto (interruptor analógico). En una computadora digital este proceso tiene lugar en un módulo de adquisición de datos o convertidor analógico-digital.

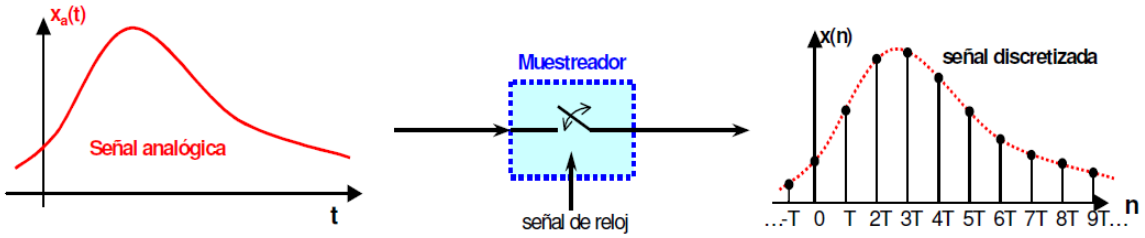


Figura 2.8 Esquema básico de un convertidor analógico-digital (A/D)

Dado que este proceso de conversión consume un tiempo significativo, cada muestra de la señal continua deberá ser "congelada" mientras dura su conversión, este congelamiento se denomina retención.

2.2.8 Teorema del muestreo y aliasing

Es evidente que al discretizar una señal de tiempo continuo se pierde algo de información en el proceso, es decir la "información" contenida en $x(n)$ no es la misma que la de la señal original $x_a(t)$, sin embargo, también es fácil ver que $x(n)$ aún contiene algo de la información de $x_a(t)$. De aquí surge en forma natural la pregunta de si es posible recuperar toda la información de la señal original $x_a(t)$ a partir de su versión discretizada $x(n)$.

El Teorema del muestreo de Nyquist-Shannon da una respuesta a una pregunta aún más específica de cuándo y cómo es posible recuperar dicha información y cuándo no.

La idea subyacente en el teorema de Nyquist-Shannon es que toda señal analógica $x_a(t)$ tiene un contenido de frecuencias, es decir, se puede expresar como una suma de sinusoides de diferentes amplitudes y frecuencias:

$$x_a(t) = \sum_{k=1}^{N} A_k \cos(2\pi F_s t + \Theta_k)$$

Por lo tanto, $x_a(t)$ contiene alguna frecuencia máxima *Fmax* que determinará la mínima frecuencia de muestreo necesaria para poder recuperar la información original.

Teorema del muestreo: Una señal de tiempo continuo $x_a(t)$ cuya máxima frecuencia contenida es *Fmax,* muestreada a una velocidad *Fs*, puede ser recuperada completamente a partir de la señal muestreada $x(n)$ si y solo si:

$$Fs > 2Fmax$$

Si se cumple esta condición la señal $x_a(t)$ se recuperará mediante la fórmula de interpolación de Shannon siguiente:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x(n) g(t - nT), \qquad \text{donde} \qquad g(t) = \frac{\sin(\pi F_s t)}{\pi F_s t}$$

,y donde $x(n)=x_a(nT)=x_a(n/Fs)$

Debido al importante papel que juega la tasa de muestreo $F_N=2Fmax$, a ésta se le llama la tasa de muestreo de Nyquist. [SyS.2]

Observaciones:

- El Teorema de Nyquist-Shannon es un resultado teórico, en el cual se considera un muestreador idealizado y no se consideran errores en los componentes electrónicos utilizados para implementar de manera práctica el muestreo o la reconstrucción de la señal. En un sistema real, la restricción teórica que establece el Teorema (*Fs>2Fmax*), deberá ser comprobada y aumentada.
- La fórmula de interpolación es no causal, de manera que no puede ser implementada en tiempo real.
- La hipótesis sobre el contenido de frecuencias de $x_a(t)$ tampoco es directamente aplicable a cualquier señal y aunque la mayoría de las señales de interés tienen un contenido de frecuencias limitado, en el caso en que no lo tengan, el teorema no se puede aplicar. Por otro lado, siempre es posible usar filtros analógicos “paso bajo” para evitar problemas con el contenido de frecuencias más altas que *2Fs*.

Si las condiciones de muestro no se satisfacen, entonces las frecuencias se pueden llegar a traslapar; es decir, las frecuencias superiores a la mitad de la frecuencia de muestreo serán reconstruidas y aparentarán ser frecuencias por debajo de la frecuencia de muestreo. El resultado sería una distorsión llamada aliasing.

Para eliminar el aliasing, los sistemas de digitalización incluyen filtros paso bajo, que eliminan todas las frecuencias que sobrepasan la frecuencia crítica (la que corresponde a la mitad de la frecuencia de muestreo elegida) en la señal de entrada. Es decir, todas las frecuencias que queden por encima de la frecuencia máxima a muestrear seleccionada, son eliminadas. El filtro paso bajo para este uso concreto recibe el nombre de filtroantialiasing.

2.3 SISTEMAS DIGITALES

2.3.1 Sistemas Digitales, solución software y hardware

Dentro de los sistemas digitales que se encuentran en el mercado se puede hacer una primera distinción entre ellos según se trate de una solución software o por el contrario de una solución hardware [ED. 3].

1. **Solución Software**. También denominados sistemas digitales microprocesados o de arquitectura fija, se encuentran dispositivos tales como:
   - Microprocesadores
   - Microcontroladores
   - Procesadores digitales de señales o *DSP* (*Digital Signal Processor*)
   - Microcontroladores de aplicación específica ASSP (*Application Specific Standard Product*)

   Todos estos dispositivos tienen como característica común que la entrada de diseño se basa en un lenguaje de alto nivel, entre ellos el más común es el *assembler*.

2. **Soluciones Hardware** o de arquitectura configurable. En este caso la entrada de diseño se basa en esquemáticos o lenguajes HDL (*Hardware Description Language*). Dentro de esta clase se distinguen los siguientes tres grandes grupos:

   2.1 Componentes discretos o circuitos integrados basados en puertas lógicas.

   La entrada de diseño se hace sobre los propios componentes o sobre la placa PCB (*Printed Circuit Board*). Este tipo de diseño presenta los siguientes inconvenientes:

   - Elevada área necesaria para el diseño.
   - Elevada complejidad de *Routing* o conexionado de los componentes. Esto hace que el diseño sea complicado y encarezca el precio final del producto.
   - Baja flexibilidad de diseño, es decir, se encuentran bastante limitados en diseños relativamente complicados.
   - Elevado consumo del dispositivo.

2.2 Lógica programable.

Se entiende por dispositivo programable aquel circuito de propósito general que posee una estructura interna que puede ser modificada, por el usuario final o por el fabricante, para implementar una amplia gama de aplicaciones.

El primer dispositivo que cumplió estas características era una memoria PROM (*Programmable Read-Only Memory*). Este componente puede tener un comportamiento de circuito utilizando las líneas de direcciones como entradas y las de datos como salidas implementando una tabla de verdad. Se pueden encontrar dos tipos básicos de PROM:

- Programables por máscara. Proporcionan mejores prestaciones y son denominadas PROM de conexiones *hardwired*.
- Programables en el campo (*field*) por el usuario final. Son las EPROM y EEPROM. Proporcionan peores prestaciones, pero son menos costosas para volúmenes pequeños de producción y se pueden programar de manera inmediata.

A este punto se le prestará especial atención puesto que dentro de este grupo se encuentran los dispositivos FPGA´s que se utilizan en este diseño. Así pues, se muestra una lista con sus ventajas e inconvenientes y se enumeran los dispositivos que componen esta clase para finalmente analizar los FPGA´s. Entre las ventajas que presentan se puede destacar que:

- Son dispositivos eficientes si implementan circuitos no superiores a unos centenares de puertas lógicas.
- Ofrecen una respuesta rápida ya que las pistas de interconexión son pequeñas y además consiguen una pequeña capacidad parásita entre pistas.
- Circuito integrado más pequeño en dimensiones que la placa PCB.
- Los recursos lógicos y la interconexión son programables por lo que se abre un gran abanico de posibilidades de programación en el diseño.

Entre los inconvenientes cabe destacar:

- Arquitectura rígida; su estructura está completamente fijada y está limitado por un número fijo de biestables y entradas /salidas.
- Fuertemente restringidos por su capacidad.

2.3 Dispositivos no programables.

ASIC (*Aplication Specific Integrated Curren*t). Son dispositivos de aplicación específica que proporcionan muy buenas prestaciones y alta capacidad de procesado. Entre sus características cabe destacar:

- Muy alta densidad de integración en cada chip.
- Muy alta velocidad de operación con señales y optimización del chip.
- Poco retardo de la señal en el proceso debido a las cortas pistas de interconexión.

2.3.2 Sistemas Lógicos Programables.

La lógica programable se basa en dispostivos PLD (*Programmable Logic Device*); los cuales se compone de una matriz de puertas AND conectada a otra matriz de puertas OR más biestables. Cualquier circuito lógico se puede implementar, por tanto, como suma de productos. Dentro de la gama de PLD se puede hacer otra distinción entre: [ED.4]

1. SPLD *(Simples Programmable Logic Device)* el cual presenta como principal característica que la estructura de interconexión es a un nivel. En este grupo se encuentran:

   - PAL (*Programmable Array Logic*): Se trata de la versión más básica dentro de los sistemas digitales programables. Se compone de un plano de puertas AND programable y otro fijo no programable de puertas OR. A continuación se muestra en la Figura 2.9:

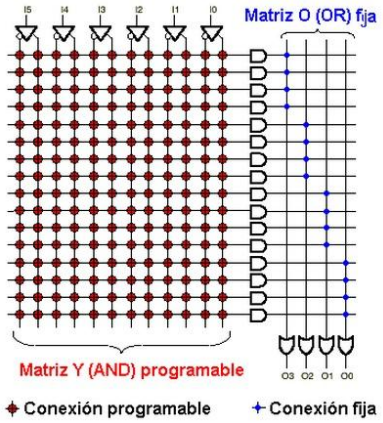


Figura 2.9 Estructura básica de un PAL.

   - GAL (*Generic Array Logic*): Se trata de un avance sobre la PAL. Es más flexible ya que se pueden programar las conexiones entre los dos planos, es decir tanto el plano AND como el OR son programables. Estos dispositivos son muy simples y producen buenos resultados con funcionalidades sencillas.

2. CPLD (*Complex Programmable Logia Device*) cuya estructura de interconexión se realiza a 2 niveles, es decir, en el circuito integrado hay presentes varias PAL´s integradas y comunicadas mediante un bus general, que provoca un plano AND complejo con una realimentación de señales (Figura 2.10). Así pues el número de puertas AND que presentan es muy superior al de las SPLD y las interconexiones son programables. Entre las características de estos dispositivos cabe destacar:
   - *Fan in*: número máximo de entradas puertas AND.
   - *Fan out:* número máximo de puertas que pueden actuar como salida.

3. FPGA

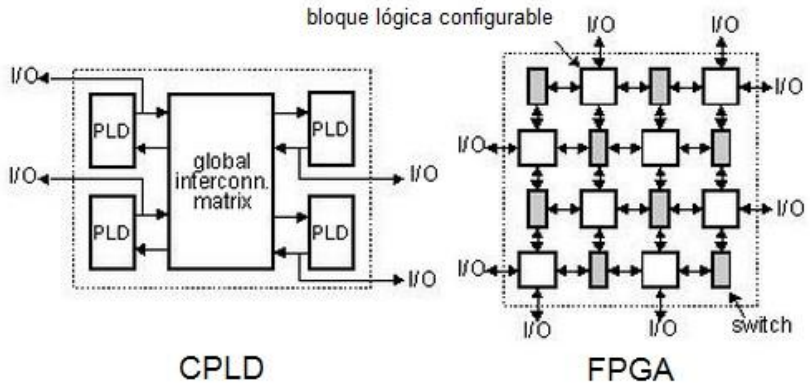


Figura 2.10 Estructura básica de CPLD y de una FPGA.

2.3.3 MPGA (*Mask-Programmable Gate Array*).

Cuyo principal representante está constituido por un conjunto de transistores más circuitería de E/S (entrada/salida), las cuales se unen mediante pistas de metal que hay que trazar de forma óptima, siendo esta la máscara que se envía al fabricante. Así pues podemos definirla como una matriz de elementos variados que se puedan interconectar libremente. Su estructura permite la implementación de circuitos complejos ya que la cantidad de interconexiones aumenta en forma proporcional a la cantidad de lógica; sin embargo, su coste inicial es elevado por la necesidad de generar la máscara de la capa de metal y fabricar el chip. Este costo se reduce con el aumento de volumen.

2.3.4 Introducción a los FPGA (*Field-Programmable Gate Array*)

En esta década, el campo de la computación está siendo dominado por sistemas empotrados, aplicaciones informáticas y procesadores de aplicación específica. Cuando los requisitos de coste, consumo de potencia, etc., no pueden ser cubiertos por los procesadores comerciales, los ingenieros normalmente se ven obligados a diseñar hardware a medida, empleando circuitos integrados de aplicación específica, conocidos en inglés como *Application Specific Integrated Circuit* (ASIC). Para construir estos sistemas los diseñadores necesitan disponer de tecnologías que permitan implementar sistemas sofisticados y de herramientas de Diseño Asistido por Computadora (CAD) que ayuden en la generación de arquitecturas que cumplan las especificaciones de cada aplicación. La metodología de diseño actual se basa en la experiencia del diseñador para implementar una arquitectura determinada en hardware.

Los diseños hardware orientados a obtener altas prestaciones han conseguido excelentes resultados debido en parte a que los ingenieros explotan manualmente la concurrencia y la especialización, lo cual permite mucha flexibilidad y la posibilidad de tener un control de grano fino sobre el diseño. Sin embargo, también es un trabajo muy tedioso y muy propenso a la introducción de errores y que requiere de un tiempo de depuración considerable.

Por otra parte, la tecnología basada en Lógica Reprogramable, conocida en ingles como *Field-Programmable Gate Array* (FPGA) ofrece una solución atractiva para un amplio rango de aplicaciones. Con su inherente programabilidad, los FPGAs muestran características normalmente asociadas a procesadores programables. Además pueden ofrecer prestaciones superiores en un orden de magnitud sobre estos. Es precisamente la combinación de flexibilidad y de prestaciones lo que las sitúa en un lugar privilegiado entre los procesadores de propósito general y los ASICs.

Hace pocos años era difícil considerar el uso de FPGAs, por ejemplo, para implementar algoritmos de Procesamiento Digital de Señal (DSP) debido a la longitud de palabra necesaria en operaciones aritméticas en punto fijo o en punto flotante. Sin embargo, esta situación ha cambiado debido a que los FPGAs han aumentando notablemente su capacidad (por ejemplo, la familia Virtex-II de Xilinx posee hasta 10 millones de puertas). Además, otros factores como los reducidos tiempos de desarrollo, entre la especificación del diseño y la salida del producto al mercado, hacen muy competitivos a estos dispositivos para soluciones DSP de aplicación específica.

En la pasada década los lenguajes de descripción hardware tales como *Very High-speed Integrated Circuit Harware Description Language* (VHDL) y Verilog, permitieron un incremento en la sofisticación de los circuitos digitales que los diseñadores eran capaces de implementar. Sin embargo, estos lenguajes requieren que el diseñador especifique el circuito en un nivel de abstracción muy bajo y además que planifique las operaciones en el hardware ciclo a ciclo. Con el incremento de la complejidad de las aplicaciones, es mayor la dificultad de explotar la concurrencia y la especialización manualmente y, por lo tanto, es indispensable disponer de herramientas que permitan un flujo de diseño automático, no sólo para lograr un diseño rápido, sino también para usar eficientemente los recursos disponibles. Por esto, en la comunidad de diseñadores se tendía a considerar que los lenguajes de especificación hardware se basan en un nivel de abstracción tan bajo que no son adecuados para especificar los grandes y sofisticados sistemas que ahora es posible implementar en hardware.

En nuestros días las librerías de generación de hardware para FPGAs han incrementado su complejidad para permitir implementaciones para procesamiento de señal y para otras aplicaciones complejas. Los algoritmos de procesamiento de señal generalmente se caracterizan por tener longitudes de palabra menores a 32 bits, posibilidad de explotar paralelismo y algoritmos de control relativamente simples que pueden ser planificados estáticamente, lo que les convierte en serios candidatos para ser implementados en los FPGAs modernos

La forma de abordar la complejidad de los diseños actuales es mediante la utilización de librerías avanzadas de diseño. Los nuevos componentes lógicos empotrados, tanto dentro de los bloques lógicos configurables como fuera de ellos, dan la oportunidad de explorar desde una nueva perspectiva el espacio de diseño. La productividad de los usuarios de FPGAs puede incrementarse enormemente si se tienen librerías de componentes que sean eficientes y fáciles de usar. Una librería parametrizable puede ser usada para generar una

amplia gama de implementaciones que admitan distintas arquitecturas, longitudes de palabra variables y comparativas velocidad-área. Tales librerías posibilitan el uso de los recursos específicos de los dispositivos permitiendo a los diseñadores obtener un buen rendimiento con una baja ocupación de recursos, y al mismo tiempo reducen la necesidad de conocimientos de detalles de bajo nivel. Una librería parametrizable proporciona módulos con las siguientes características:

a) implementados óptimamente
b) con una variedad amplia de elementos configurables que se adapte a las necesidades del usuario.
c) que sea fácil de usar y que se adapte al flujo de diseño de otras herramientas.
d) que este cuidadosamente validada.

2.3.5 FPGA

Introducidos por Xilinx en 1985, también se denominan LCA (*Logic Cell Array*). Los FPGAs son miembros de un grupo de dispositivos llamados Dispositivos programables en campo, conocidos en inglés como *Field-Programmable Devices* (FPD). Se pueden definir como un circuito integrado que contiene una matriz de celdas lógicas que pueden ser programadas por el usuario para actuar como un circuito digital a la medida. Pueden ser utilizados para cualquier aplicación en la que son utilizados los ASICs. Por ejemplo, se puede implementar en un FPGA funciones digitales de decenas de puertas lógicas como multiplexores, decodificadores, sumadores, máquinas de estado, memorias, microprocesadores, funciones de procesamiento digital tales como filtros y transformadas o incluso, sistemas completos en un solo circuito integrado, conocidos en inglés como *System On Chip* (SOC). Los FPGAs tienen un coste de desarrollo de producto relativamente bajo, un ciclo de diseño corto, y pueden ser reprogramadas muchas veces, incluso para corregir pequeños fallos de diseño en los productos finales.

2.3.6 Arquitectura de los FPGAs

Los FPGAs son los dispositivos programables más versátiles y poderosos disponibles hoy en día. Los elementos principales de un FPGA son la matriz de bloques lógicos, los bloques de I/O y las líneas de interconexión [FPGA.1].

Bloques lógicos

Un bloque lógico usualmente contiene generadores de funciones lógicas Look-Up Table (LUT) y elementos de almacenamiento (ip-ops). Las LUTs son usadas para implementar funciones lógicas combinacionales normalmente de cuatro entradas y una salida. Los elementos de almacenamiento pueden ser usados como ip-ops o latches. Además de los elementos mencionados, los bloques lógicos contienen recursos especiales para manejo de operaciones aritméticas, multiplexores extras para enlazar varios bloques lógicos en mejores condiciones y líneas de interconexión rápidas para el manejo de señales de reloj, de acarreo o de reset.

Líneas de interconexión

Una FPGA contiene líneas de interconexión flexibles para conectar los diversos bloques lógicos. Hay varios tipos de líneas de conexión en una FPGA

a) líneas largas para conectar bloques distantes
b) líneas cortas para conectar bloques lógicos adyacentes
c) líneas rápidas para rutas críticas en circuitos aritméticos
d) líneas de set y reset para los ip-ops
e) árboles de reloj usados para distribuir señales de sincronización

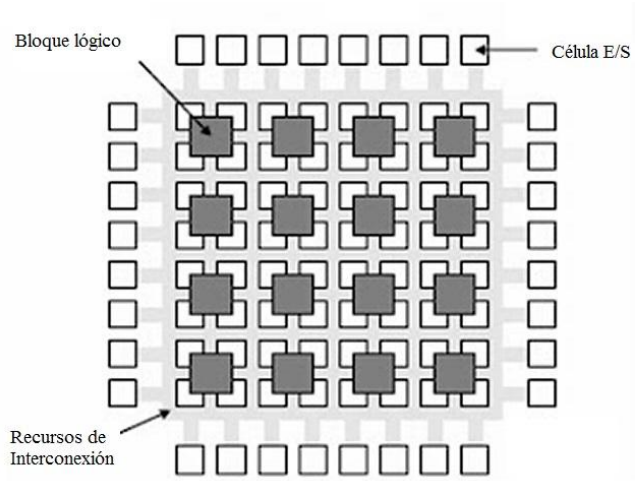


Figura 2.11 Estructura interna de un FPGA donde se muestra: bloques lógicos, bloques de entrada/salida y líneas de interconexión

Otros recursos

Las FPGAs modernas no limitan sus recursos a los mencionados anteriormente y contienen otros elementos de alto nivel fijos en silicio tales como (Figura 2.12):

- Bloques de memoria que pueden ser utilizados como ROM, RAM y FIFOs, pudiendo eliminar la necesidad de chips de memoria externos.
- Bloques de manejo de reloj, que son utilizados para sintetizar señales de reloj.
- Módulos DSPs que contienen hardware aritmético y que son utilizados en bloques de filtros y en otros algoritmos.
- CPUs que en caso del fabricante Xilinx, proporciona varios procesadores PowerPC.
- Transceptores serie especializados, que permiten entradas/salidas de alta velocidad en varios estándares de comunicación serie.

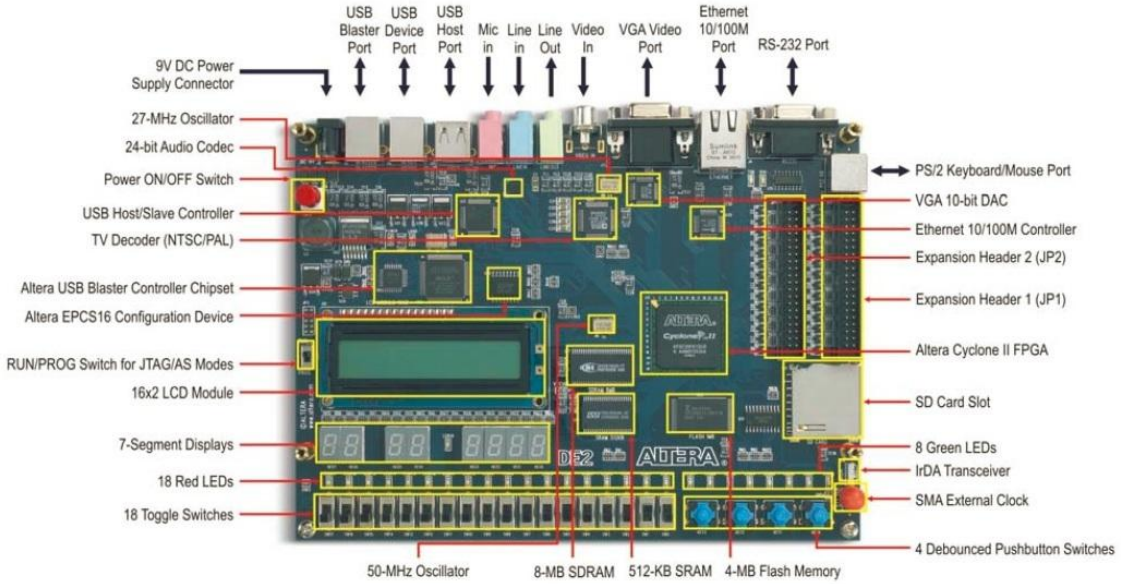


Figura 2.12 Kit DE2 Altera

2.3.7 Ventajas y desventajas de los FPGAs

Entre las ventajas de utilizar FPGAs frente a otros dispositivos digitales están:

1. El tiempo de programación y puesta en el mercado se reduce considerablemente.
2. Son programables por el usuario. Esto a parte de dar mayor libertad al diseñador permite una reducción de productos en stock ya que se pueden aplicar para diferentes aplicaciones.
3. Algunos tipos son reprogramables, lo que los hace especialmente adecuados para procesos de prototipo en muchos diseños y permite la corrección de errores.
4. El proceso de diseñó es simple y asequible.
5. Existe una amplia gama de fabricantes que cubren las diferentes necesidades de cada usuario.

Como inconvenientes se tiene que:

1. Es un dispositivo más lento que otro de propósito específico debido principalmente a los transistores y a la matriz de interconexión que utiliza, por hacer una idea aproximada los mecanismos de interconexión de un FPGA introduce entre el 30% y el 50% del retardo total de todo el circuito.
2. En ocasiones se desaprovecha parte de la lógica para poder realizar el rutado del sistema.

2.3.8 Arquitectura del FPGA Cyclone II de Altera

EL fabricante Altera, diferencia las CPLDs y los FPGAs por diferentes estructuras de interconexión. La estructura segmentada es utilizada por los FPGAs; los cuales utilizan líneas múltiples de longitud variable unidas por transistores de paso o “antifusibles” para conectar las celdas lógicas. En contraste, la estructura de interconexión continua es

utilizada por las CPLDs para proveer conexiones de celda lógica a celda lógica que lleva finalmente a velocidades más altas comparándolas con los FPGAs.

Cada dispositivo FPGA de Altera está integrado por bloques de memoria y una matriz de elementos lógicos. Los bloques de memoria son conocidos como EABs (*Embeded Array Blocks*) y pueden utilizarse para definir pequeñas memorias o funciones lógicas especiales. En algunas FPGAs más actuales estos bloques de memoria son denominados (*Embedded System Blocks*), los cuales incorporan nuevas funciones.

La matriz de elementos lógicos (*Logic Array*) está constituida por bloques lógicos conocidos como LABS (*Logic Array Blocks*) (véase Figura 2.13 a)). Cada LAB agrupa elementos lógicos o LEs (*Logic Elements*) e interconexiones locales. Un LE es una unidad pequeña de lógica que proporciona una aplicación eficaz de funciones lógicas. Un LE incluye tablas de look-up (LUT), flip-flop programable y lógica para la rápida generación y propagación de acarreo (array).

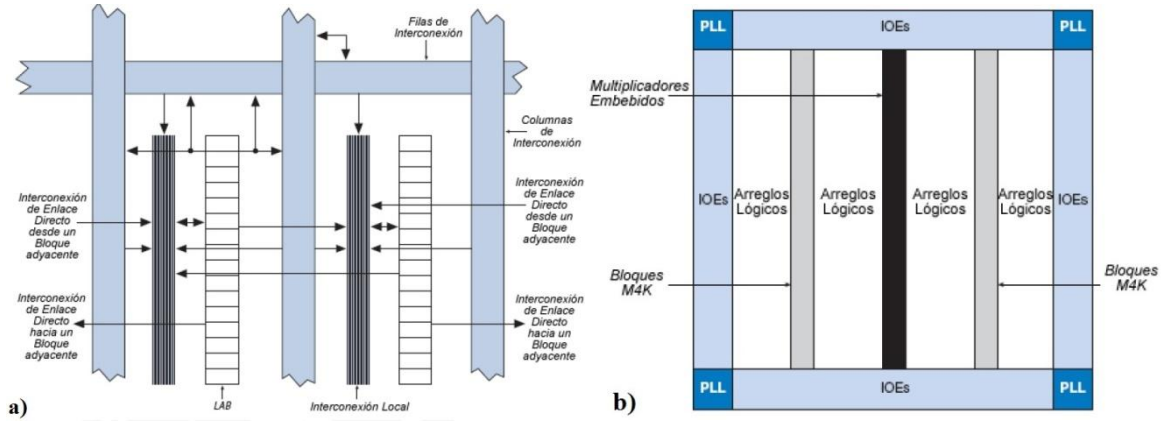


Figura 2.13 a) Estructura del LAB **b)** Diagrama de bloques del FPGA Cyclone II

Las conexiones entre elementos de memoria y elementos lógicos se realizan mediante el llamado *FastTrack Interconnect*, una serie de rápidos canales de fila y columna que recorren todo el ancho y el alto del dispositivo.

Cada pin de Entrada/Salida se alimenta por un elemento de Entrada/Salida (IOE, *Input/Output Element*) localizado al final de cada fila y columna del *FastTrack Interconnect*.

Los dispositivos Cyclone contienen una arquitectura basada en fila y columna de dos dimensiones para implementar lógica. Las interconexiones columna y fila de distintas velocidades proporcionan señales de interconexión entre los LABs y los bloques integrados de memoria. [FPGA.2]

2.3.9 Kit DE2 Altera y FPGA Cyclone II

Para el proyecto se ha utilizado el Kit DE2 del fabricante Altera (Figura 2.12); el cual lleva embebido un FPGA del modelo Cyclone II EP2C35F672C6 con 35k LEs; correspondiente a un circuito integrado de la familia Cyclone^{TM}II. Esta pastilla está compuesta por 35000 elementos lógicos, un bloque de memoria de 484kb y 475 pines de entrada/salida.

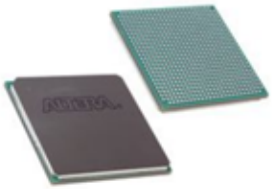


Figura 2.14 Aspecto del encapsulado de la pastilla CYCLONE II FPGA de Altera

2.3.10 Aplicaciones de los FPGAs

Las FPGAs se aplican actualmente en el procesamiento digital de señal, sistemas aeroespaciales y de defensa, prototipado de ASICs, visión por computadora, reconocimiento de voz, criptografía, bioinformática, emulación de hardware de computadora y redes neuronales entre otras.

Los FPGAs empezaron como competidores de las CPLDs para el reemplazo de circuitos digitales discretos. Sin embargo, con el incremento de sus capacidades, tamaño y velocidad, los FPGAs comenzaron a abarcar funciones cada vez más complejas, hasta llegar a implementar en nuestros días SOCs (*System on Chip).* Particularmente, con la introducción de multiplicadores empotrados, empezaron a instalarse en sistemas reservados especialmente para los chip DSP.

También los FPGAs han incrementado su uso en aplicaciones convencionales de computación de altas prestaciones, conocida en inglés como *High Performance Computing* (HPC), donde funciones matemáticas como la Transformada Rápida de Fourier, FFT (*Fast Fourier Transform*) es realizada en FPGAs en vez de microprocesador. Tradicionalmente la adopción de FPGAs en HPC (*High-performance computing*) es limitada por la complejidad del diseño con FPGAs comparada con el diseño de software convencional. En la medida de que haya disponibles mejores herramientas de diseño para FPGAs esta situación irá cambiando.

2.3.11 Flujo de diseño con FPGAs

El flujo de diseño con FPGAs normalmente utiliza los siguientes pasos:

- Diseño de arquitectura. Que implica el análisis de los requerimientos del problema y de la estructura que cumple con dichos requerimientos. Aquí pueden evaluarse diversas opciones para seleccionar la que más convenga de acuerdo a las especificaciones. También se definen los bloques estructurales, sus interfaces y sus funciones.

- Descripción en HDL. El diseño es descrito formalmente en un lenguaje de descripción de hardware, normalmente VHDL o Verilog.

- Simulación funcional. Es deseable verificar que la descripción de hardware corresponde con el circuito que se desea diseñar. Normalmente las herramientas de diseño disponen de un bloque de simulación en el que es posible dar vectores de prueba a circuito descrito con HDL.

- Síntesis. Esta etapa consiste en trasladar el código HDL en una lista de conexiones, llamada en inglés *netlist*. La síntesis implica también que el diseño especificado cumpla con algunos criterios de diseño, como área y velocidad. Si los criterios no son cumplidos, se pueden ajustar los parámetros de síntesis de la herramienta para producir una mejor optimización. Si aún con los ajustes de la herramienta, no es posible cumplir con los criterios de diseño, entonces debe cambiarse la descripción del hardware en el código HDL y repetir el proceso.

- Implementación. Consiste en colocar cada una de las expresiones lógicas obtenidas en el proceso de síntesis, en los recursos contenidos internamente del FPGA destino (*Map*). Posteriormente el software realiza el emplazamiento y rutado de los recursos seleccionados (*Place and Route*).

- Configuración. Una vez que el diseño cumple con los requerimientos planteados y que dicho diseño es empotrable en el FPGA, puede descargarse a través de software y hardware de configuración al chip destino.

2.4 VHDL (*Very High Description Lenguage*)

2.4.1 Lenguajes de descripción hardware

Los lenguajes de descripción hardware (HDLs, *Hardware Description Languages*) vienen utilizándose desde los años 70 en los ciclos de diseño de sistemas digitales asistidos por herramientas de CAD electrónico. Al principio surgieron una serie de lenguajes que no llegaron a alcanzar un éxito que permitiera su consolidación en el campo industrial o académico. En los años 80 aparecen los lenguajes Verilog y VHDL que, aprovechando la disponibilidad de herramientas hardware y software cada vez más potentes y asequibles y los adelantos en las tecnologías de fabricación de circuitos integrados, lograron imponerse como herramientas imprescindibles en el desarrollo de nuevos sistemas. En la actualidad ambos lenguajes están normalizados y comparten una posición hegemónica que está arrinconando y terminará, probablemente en poco tiempo, eliminando del mercado al resto de lenguajes que de un modo u otro todavía son soportados por algunas herramientas de CAD.

Estos lenguajes son sintácticamente similares a los de programación de alto nivel. Verilog tiene una sintaxis similar al C y VHDL a ADA y se diferencian de éstos en que su semántica está orientada al modelado del hardware. Su capacidad para permitir distintos enfoques en el modelado de los circuitos y su independencia de la tecnología y metodología de diseño permiten extender su uso a los distintos ciclos de diseño que puedan utilizarse. Por ello, para los profesionales relacionados de alguna manera con el diseño o mantenimiento de sistemas digitales resulta hoy en día imprescindible su conocimiento.

2.4.2 El lenguaje VHDL

Los estudios para la creación del lenguaje VHDL (*Very High Speed Integrated Circuit* VHSIC HDL) comenzaron en el año 1981, bajo la cobertura de un programa para el desarrollo de Circuitos Integrados de Muy Alta Velocidad (VHSIC), del Departamento de Defensa de los Estados Unidos. En 1983 las compañías Intermetrics, IBM y Texas Instruments obtuvieron la concesión de un proyecto para la realización del lenguaje y de un conjunto de herramientas auxiliares para su aplicación. Finalmente, en el año 1987, el lenguaje VHDL se convierte en la norma IEEE-1076; como todas las normas IEEE, se somete a revisión periódica, por lo que en 1993 sufrió algunas leves modificaciones [VHDL.1].

2.4.3 Características del lenguaje

El lenguaje VHDL fue creado con el propósito de especificar y documentar circuitos y sistemas digitales utilizando un lenguaje formal. En la práctica se ha convertido, a traves de un gran número de entornos de CAD, en el lenguaje HDL de referencia para realizar modelos sintetizables automáticamente. Las principales características del lenguaje VHDL se explican en los siguientes puntos:

- Descripción textual normalizada: El lenguaje VHDL es un lenguaje de descripción que especifica los circuitos electrónicos en un formato adecuado para ser interpretado tanto por máquinas como por personas. Se trata además de un lenguaje formal, es decir, no resulta ambiguo a la hora de expresar el comportamiento o representar la estructura de un circuito. Está, como ya se ha dicho, normalizado, o sea, existe un único modelo para el lenguaje, cuya utilización está abierta a cualquier grupo que quiera desarrollar herramientas basadas en dicho modelo, garantizando su compatibilidad con cualquier otra herramienta que respete las indicaciones especificadas en la norma oficial.
  Es por último, un lenguaje ejecutable, lo que permite que la descripción textual del hardware se materialice en una representación del mismo utilizable por herramientas auxiliares tales como simuladores y sintetizadores lógicos, compiladores de silicio, simuladores de tiempo, de cobertura de fallos, herramientas de diseño físico, etc.

- Amplio rango de capacidad descriptiva**:** El lenguaje VHDL posibilita la descripción del hardware con distintos niveles de abstracción, pudiendo adaptarse a distintos propósitos y utilizarse en las sucesivas fases que se dan en el desarrollo de los diseños. Además es un lenguaje adaptable a distintas metodologías de diseño y es independiente de la tecnología, lo que permite, en el primer caso, cubrir el tipo de necesidades de los distintos géneros de instituciones, compañías y organizaciones relacionadas con el mundo de la electrónica digital; y, en el segundo, facilita la actualización y adaptación de los diseños a los avances de la tecnología en cada momento.

- Otras caracterísitcas: Además de las ventajas ya reseñadas también es destacable la capacidad del lenguaje para el manejo de proyectos de grandes dimensiones, las garantías que comporta su uso cuando durante el ciclo de mantenimiento del proyecto, hay que sustituir componentes o realizar modificaciones en los circuitos, y el hecho de que, para muchas organizaciones contratantes, sea parte indispensable de la documentación de los sistemas.

2.4.4 Estilos de Descripción

La descripción puede hacerse, fundamentalmente, de dos formas:

1. Mediante construcciones que representan un algoritmo de procesamiento.
2. Mediante la interconexión de otros dispositivos.

Por ejemplo, siguiendo la primera metodología, para describir el funcionamiento de una puerta AND de tres entradas, puede utilizarse una sentencia como:

sal <= ent1 AND ent2 AND ent3

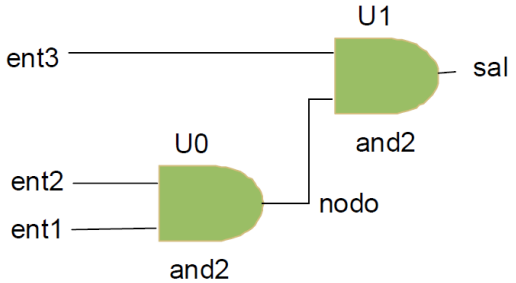


Figura 2.15 Esquema físico lógico

De acuerdo con la segunda opción, también podría utilizarse el circuito de la Figura 2.15, en el que se emplean puertas AND de dos entradas.

Cuando se opta por la descripción del algoritmo de procesamiento pueden emplearse distintos niveles de abstracción. Los distintos enfoques con que puede abordarse el modelado del funcionamiento de los circuitos se denominan estilos. Este concepto no se trata en la norma del lenguaje; procede de su aplicación práctica.

Una descripción en la que el funcionamiento se describe mediante la interconexión de otros dispositivos se denomina estructural y es la de menor grado de abstracción; a continuación viene el estilo denominado RTL (*Register Transfer Level*), en el que se utilizan construcciones simples (ecuaciones lógicas, operaciones aritméticas básicas), muy cercanas a la lógica descrita, de las que resulta fácil deducir la estructura hardware modelada. Son las descripciones más abstractas que "entienden" las herramientas de síntesis lógica automática. El mayor grado de abstracción corresponde a las descripciones de comportamiento (*behavioural*); en este estilo, el procesamiento realizado por el hardware se describe mediante algoritmos tan complejos como los que pueden realizarse con los lenguajes de programación de alto nivel. Suele utilizarse para la evaluación e investigación de arquitecturas de sistemas digitales y, durante el ciclo de diseño para facilitar la simulación de circuitos en fase de desarrollo. En los últimos años han aparecido algunas herramientas de "síntesis arquitectural" que admiten este estilo, con fuertes restricciones, como código de entrada.

Los distintos niveles de abstracción pueden adaptarse a distintos propósitos y utilizarse en las sucesivas fases que se dan en el ciclo de diseño de los circuitos digitales. Por ejemplo, si se está investigando la validez de una arquitectura para la realización de un determinado algoritmo de procesamiento resulta sencillo y rápido modelar el hardware con un estilo de comportamiento. Si se decide diseñar el sistema con la ayuda de un sintetizador lógico, se debe traducir la descripción a un estilo, RTL, inteligible por la herramienta de síntesis, que tendrá condicionado el empleo de determinados elementos del lenguaje.

Finalmente, el software que realiza la síntesis lógica puede generar una descripción estructural (*netlist*) en lenguaje VHDL. Para optimizar la eficacia en el uso del lenguaje debe emplearse siempre el mayor nivel de abstracción posible, ya que el número de líneas que ocupa una descripción y el tiempo que se tarda en realizarla, así como las dificultades que presenta la modificación del modelo, crecen cuanto más "cerca" se está del hardware.

Las descripciones estructurales no tienen misterios: para realizarlas basta con conocer las normas del lenguaje que indican como conectar modelos de dispositivos en un Cuerpo de Arquitectura; la tarea de realizar una descripción estructural puede resultar tediosa pero es sencilla, además en muchos entornos de CAD es posible generar automáticamente este tipo de descripciones a partir de esquemas gráficos. Las descripciones RTL para síntesis son más complicadas porque obligan a seguir un conjunto de reglas de construcción, dependientes en alguna medida de la herramienta de síntesis que se vaya a usar. A pesar de ello existe un conjunto de patrones de codificación (cómo y cuáles construcciones utilizar)

que resultan válidos para la inmensa mayoría de las herramientas de síntesis y permiten esbozar unas reglas básicas de codificación. Por su parte, las de comportamiento se realizan utilizando las mismas construcciones del lenguaje que las RTL, pero sin otras restricciones que no sean las sintácticas y semánticas del lenguaje.

Por último hay que decir que en la práctica los modelos VHDL combinan los diferentes estilos. Basta con que el hardware sea moderadamente complejo para que existan elementos que describan aspectos estructurales y siempre habrá dispositivos básicos que no admitan ser descritos a través de la interconexión de otros.

2.5 TRANSFORMADA DISCRETA DE FOURIER (DFT)

2.5.1 Introducción

La Transformada Discreta de Fourier DFT (*Discrete Fourier Transform*) juega un papel importante en numerosas aplicaciones de procesamiento de señales digitales. Entre otras destacan: filtrado lineal, análisis de correlación y análisis espectral. Una de las razones de la importancia de la DFT es la existencia de algoritmos sumamente eficientes para su cálculo. Uno de los más utilizados es el denominado Radix 2, el cual se basa en el hecho de analizar una secuencia que posee un número de tamaño N que es una potencia de 2. Existe un algoritmo denominado Radix 4, cuyo nombre deriva de la misma razón del Radix 2. Estos tipos de algoritmos permiten calcular en forma rápida y sumamente eficiente la DFT.

La DFT requiere de grandes cantidades de operaciones matemáticas para lograr el mismo resultado que los algoritmos FFT. En particular, el cálculo de la FFT (con el algoritmo Radix 2) usando 1024 puntos puede mejorar la rapidez hasta en alrededor de 200 veces la velocidad de cálculo directo la DFT, (de N^2 a $(N/2)*\log_2 N$)

En las siguientes secciones se muestra el algoritmo FFT Radix 2 que utiliza el criterio de Diezmado en Frecuencia, el cual se basa en la descomposición sucesiva de la entrada en subsecuencias cada vez más pequeñas, hasta un límite.

El algoritmo de la FFT emplea dos técnicas para lograr cálculos rápidos y reducir el número de multiplicadores, las cuales se describen a continuación:

- Técnica No. 1: Divide y vencerás.

  Se basa en el principio fundamental de la descomposición de los cálculos de la TDF en una secuencia de longitud N en TDFs cada vez más pequeñas. La forma en que se aplica este principio nos lleva a una diversidad de diferentes algoritmos, los cuales introducen mejoras en la velocidad de cómputo.

  Esta técnica puede ser dividida en tres pasos principalmente:
  - Dividir la secuencia de datos en dos o más subsecuencias de menor tamaño.
  - Resolver cada secuencia recursivamente por el mismo algoritmo.

- Obtener la solución para el problema original mediante la combinación de las soluciones a las subsecuencias.

- Técnica No. 2: Propiedades de los coeficientes ($W_N = e^{-jk\frac{2\pi}{N}n}$).

  Para mejorar la eficiencia de los cálculos de la TDF, se aplican las propiedades siguientes:

  - $W_N^0 = -W_N^{N/2} = 1, W_N^{N/4} = -W_N^{3N/4} = -j$.
  - $W_N^{n+(k+N/2)} = -W_N^{nk}$ Simetría.
  - $W_N^{nk} = -W_N^{(n+N)k} = w_N^{n(N+k)}$ Periodicidad en n y en k.

2.5.2 Transformada Discreta de Fourier – DFT

Básicamente, el problema de cálculo de la DFT es obtener la secuencia {X(n)} de longitud *N* según la fórmula dada en *(1)*:

$$X[k] = \sum_{n=0}^{N-1} x[n] W^{nk} \qquad (1)$$

Donde $N=2^a$, *a = 1, 2, 3, ...*

El factor $W_N = e^{-2j(2\pi/N)}$ con $e^{-jx} = \cos(x) - jsen(x)$ es denominado *Twiddle Constant*.

Con lo anterior, el cálculo de un punto de la transformada discreta de Fourier está dado por:

$$X[k] = x(0)W^0 + x(1)W^k + x(2)W^{2k} + ... + x(N-1)W^{k(N-1)}, k = 0,1,2,...N-1 \qquad (2)$$

Desarrollando *(2)* para los *N* valores posibles de *k* se obtiene una matriz de tamaño *NxN*. De *(2)* se puede calcular el número de operaciones necesarias para realizar la transformación de los datos mediante este algoritmo. El número de sumas complejas que se deben realizar es de *(N-1)xN* y la cantidad de multiplicaciones complejas asciende a N^2. Es claro que esta cantidad de operaciones es alta y requiere de un enorme poder de cálculo. El cálculo directo de la DFT no es eficiente debido, fundamentalmente, a que no explota las propiedades de simetría y periodicidad del factor de fase W_N.

De la observación de *(2)* es claro que no es necesario realizar las N^2 multiplicaciones ya que los valores de los factores $W^0 = 1$ no son necesarios de multiplicar. Además existen propiedades de periodicidad y simetría en estos factores de forma tal que:

$$W^{k+N} = W^k \qquad (3)$$

$$W^{k+N/2} = -W^k \qquad (4)$$

La simetría y periodicidad de los factores *W* quedan de manifiesto en la Figura 2.16.

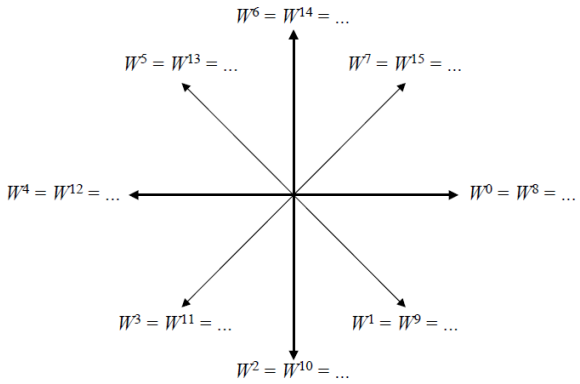


Figura 2.16 Periodicidad y simetría de los factores *W* para N=8

A partir del algoritmo DFT y las consideraciones anteriores es posible llegar a un método de cálculo mucho más eficiente, que entrega los mismos resultados y con un número menor de operaciones. Es el llamado algoritmo de Transformada Rápida de Fourier (FFT, *Fast Fourier Transform*).

2.5.3 Desarrollo del algoritmo FFT RADIX – Diezmado en frecuencia

Sea la secuencia de datos *{x(n)}*, con *n: [0 : N-1]*. Separando *{x(n)}* en dos secuencias para *n*'s pares e impares es posible aplicar la DFT a la secuencia de datos *{x(n)}*, quedando de la siguiente forma:

$$X[k] = \sum_{n=0}^{(N/2)-1} x[n] W^{nk} + \sum_{n=N/2}^{N-1} x[n] W^{nk} \qquad (5)$$

Si se reemplaza $n=n+N/2$ en la segunda sumatoria queda:

$$X[k] = \sum_{n=0}^{(N/2)-1} x[n] W^{nk} + W^{kN/2} \sum_{n=0}^{N/2-1} x\left[n+\frac{N}{2}\right] W^{nk} \qquad (6)$$

Usando $W^{kN/2}=(\cos(\pi)-j\,\mathrm{sen}(\pi))^k=-1^k$, *(6)* queda de las forma siguiente:

$$X[k] = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1) x(n+\frac{N}{2}) \right] W^{nk} \qquad (7)$$

A partir del hecho de que $(-1)^k=1$ para *k* par y $(-1)^k=-1$ para *k* impar, *(7)* puede ser separada en sumatoria para secuencias pares e impares y reemplazando *k=2k* para la sumatoria de los pares y *k=2k+1* para la sumatoria de los impares, *(7)* puede ser reescrita como

$$X[2k] = \sum_{n=0}^{(N/2)-1} \left[x(n) + x(n+\frac{N}{2}) \right] W^{2nk} \qquad \text{para } k \text{ par} \qquad (8)$$

$$X[2k-1] = \sum_{n=0}^{(N/2)-1} \left[x(n) - x(n+\frac{N}{2}) \right] W^{2nk} W^{n} \qquad \text{para } k \text{ impar} \qquad (9)$$

Para facilitar la escritura y aprovechando que los factores *W* son funciones del período *N*, los factores pueden ser reescritos de la forma siguiente: W_N. Luego W_N^2 puede ser representado como $W_{N/2}$. Sean

$$a(n) = x(n) + x(n+N/2) \qquad (10)$$

$$b(n) = x(n) - x(n+N/2) \qquad (11)$$

Las ecuaciones *(10)* y *(11)* pueden ser escritas más claramente como 2 DFT de *N/2* puntos,

$$X[2k] = \sum_{n=0}^{(N/2)-1} a[n] W_{N/2}^{nk} \qquad (12)$$

$$X[2k+1] = \sum_{n=0}^{(N/2)-1} b[n] W_{N/2}^{nk} W_N^n \qquad (13)$$

La Figura 2.17 muestra la descomposición de una DFT de *N* puntos en dos DFT de *N/2* puntos para el caso de *N = 8*. Aplicando *(12)* y *(13)* es posible llegar a obtener los *X*'s de la primera etapa de la FFT.

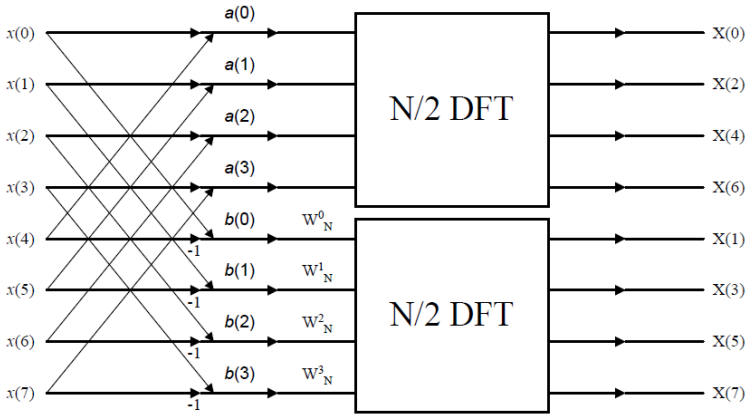


Figura 2.17 Descomposición de una DFT de *N* puntos en 2 DFTs de *N/2* puntos, para *N=8*

Como resultado del proceso de descomposición, los *X*'s quedan ordenados en grupos de pares e impares. El proceso de descomposición puede ser repetido nuevamente pero esta vez para *N/4*, que para el caso de 8 puntos, es la etapa final. El número de etapas, o DFTs, se deberá repetir hasta llegar a la DFT de 2 puntos. En general una FFT de *N* puntos tendrá a etapas, con $N=2^a$.

En la Figura 2.18 es posible ver que la salida del algoritmo está desordenada. Esto se debe a que en las etapas que componen el algoritmo cada vez se realiza un agrupamiento de los datos almacenados en posiciones pares de memoria y los almacenados en posiciones impares. Por esta razón la salida necesita ser reordenada. El proceso que deja

correctamente ordenados los datos de salida se denomina *bit-reverse* y será explicado más adelante.

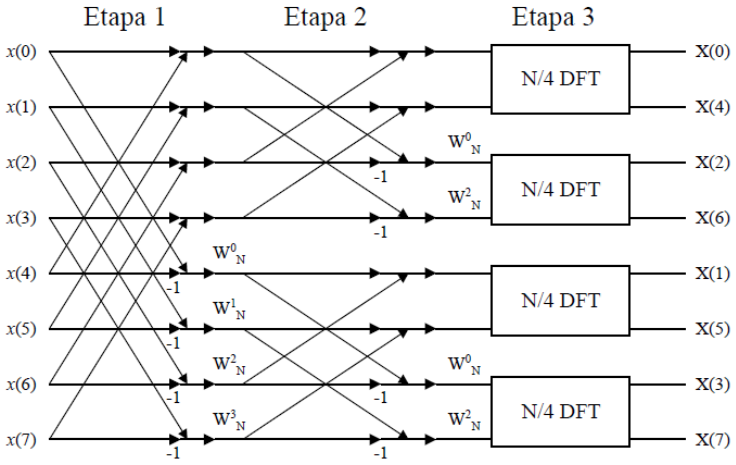


Figura 2.18 Descomposición de una DFT de *N/2* puntos en 4 DFTs de *N/4* puntos, para *N=8*

La última descomposición a la que se ha llegado al aplicar la DFT es de 2 puntos; es la más baja descomposición del algoritmo Radix 2. Una DFT de 2 puntos las salidas *X(n)* pueden ser escritas como se observa en *(14)* y (15)

$$X[0] = x(0)W^0 + x(1)W^0 = x(0) + x(1) \quad (14)$$

$$X[1] = x(0)W^0 + x(1)W^1 = x(0) - x(1) \quad (15)$$

Resaltar que $W^1 = -1$

Las ecuaciones *(14)* y *(15)* pueden ser representadas en un diagrama de flujo conocido como "Mariposa". Este es el diagrama de flujo de la etapa final de todo algoritmo FFT que utiliza el diezmado en frecuencia.

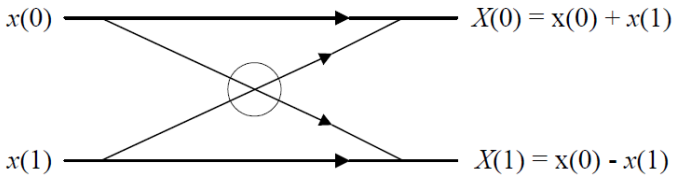


Figura 2.19 Diagrama de flujo de una FFT de 2 puntos - *Mariposa*

EL diezmado en frecuencia adquiere su nombre del hecho de que la secuencia de salida *X(k)* es descompuesta (diezmada) en subsecuencias más pequeñas, continuando por etapas o iteraciones. La salida *X(k)* posee componentes tanto reales como imaginarias, y el algoritmo FFT se puede acomodar a entradas tanto reales como complejas. La FFT no es una aproximación de la DFT, sino que es un algoritmo más eficiente que se vuelve más y más importante a medida que *N* aumenta. El diagrama final del algoritmo Radix 2, con diezmado en frecuencia es mostrado en la Figura 2.20.

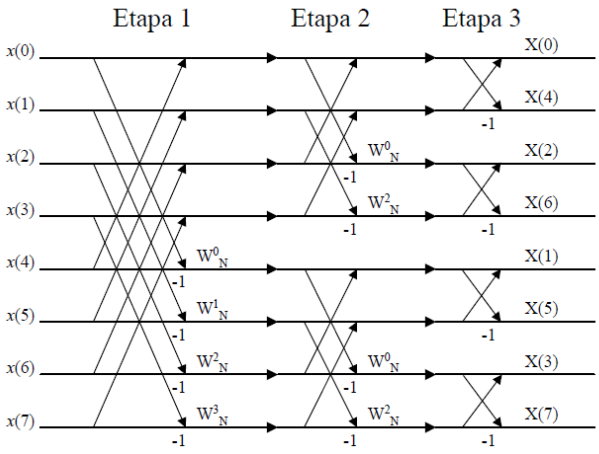


Figura 2.20 Diagrama de flujo de cálculo de una FFT de 8 puntos usando diezmado en frec.

Una alternativa al diagrama de flujo mostrado en la Figura 2.20 puede ser obtenida con entradas desordenadas (aplicando bit reverse a la secuencia de entrada) y salidas ya ordenadas.

2.5.4 Algoritmo *bit reversal* para ordenamiento de datos

Este algoritmo permite que una secuencia de entrada o salida del cálculo de la FFT sea reordenada para obtener un resultado deseado.

Para ilustrar este algoritmo se verá el caso de $N = 8$, representado por tres bits. El bit 3 y el 1 deben ser intercambiados de sus posiciones. Ejemplo, *(100)b* queda *(001)b*. Esto hace que el dato que estaba en la posición de memoria 4 = *(100)b* pase a la posición 1 = *(001)b*. Similarmente, *(110)b* es intercambiado por *(011)b*. De esta forma, una secuencia puede ser ordenada tanto para aplicarla a la entrada del algoritmo DIT o a aplicarla a la salida del algoritmo DIF. Lo anterior se resume gráficamente en la Figura 2.21:

n2n1n0	**n0n1n2**
000	000
001	100
010	010
100	001
101	101
110	011
111	111

Figura 2.21 Resultado del intercambio de bit para aplicar el reordenamiento de la salida del algoritmo FFT. Caso de 3 bits o FFT de 8 puntos

Referencias bibliográficas ap. 2.5 [FFT.1],[FFT.2][FFT.3]

2.6 SVGA (*Super Video Graphics Array*)

Super Video Graphics Array o SVGA es un término que cubre una amplia gama de estándares de visualización gráfica de ordenadores, incluyendo tarjetas de video y monitores.

2.6.1 Evolución

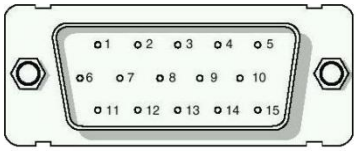
Cuando IBM lanzó al mercado el estándar VGA en 1987 muchos fabricantes manufacturaron tarjetas VGA clones. Luego, IBM crea el estándar XGA, el cual no fue seguido por las demás compañías, que comenzaron a crear tarjetas gráficas SVGA.

Las nuevas tarjetas SVGA de diferentes fabricantes no eran exactamente igual a nivel de hardware, lo que las hacía incompatibles. Los programas tenían dos alternativas: manejar la tarjeta de vídeo a través de las llamadas estándar, lo cual era muy lento pero había compatibilidad con las diferentes tarjetas, o manejar la tarjeta directamente, lo cual era muy rápido y se podía acceder a toda la funcionalidad de ésta (modos gráficos, etc.). Sin embargo, el programador tenía que hacer una rutina de acceso especial para cada tipo de tarjeta [SVGA.1].

Poco después surgió *Video Electronics Standards Association* (VESA), un consorcio abierto para promover la interoperabilidad y definición de estándares entre los diferentes fabricantes. Entre otras cosas, VESA unificó el manejo de la interface del programa hacia la tarjeta, también desarrolló un bus con el mismo nombre para mejorar el rendimiento entre el ordenador y la tarjeta. Unos años después, este bus sería sustituido por el PCI de Intel. SVGA fue definido en 1989 y en su primera versión se estableció para una resolución de 800 × 600 píxeles y 4 bits de color por píxel, es decir, hasta 16 colores por píxel. Después fue ampliado rápidamente a 1024 × 768 píxeles y 8 bits de color por píxel, y a otras mayores en los años siguientes. Aunque el número de colores fue definido en la especificación original, esto pronto fue irrelevante, (en contraste con los viejos estándares CGA y EGA), ya que el interfaz entre la tarjeta de vídeo y el monitor VGA o SVGA utiliza voltajes simples para indicar la profundidad de color deseada. En consecuencia, en cuanto al monitor se refiere, no hay límite teórico al número de colores distintos que pueden visualizarse, lo que se aplica a cualquier monitor VGA o SVGA.

Mientras que la salida de VGA o SVGA es analógica, los cálculos internos que la tarjeta de vídeo realiza para proporcionar estos voltajes de salida son enteramente digitales. Para aumentar el número de colores que un sistema de visualización SVGA puede producir, no se precisa ningún cambio en el monitor, pero la tarjeta vídeo necesita manejar números mucho más grandes y puede ser necesario rediseñarla desde el principio. Debido a esto, los principales fabricantes de chips gráficos empezaron a producir componentes para tarjetas vídeo del alta densidad de color apenas unos meses después de la aparición de SVGA.

El conector VGA común (otros nombres incluyen conector RGBHV, D-sub 15, sub mini D15 y D15) contiene 15 pines DE-15 en sus tres hileras (véase Figura 2.22). Éste se encuentra en la mayoría de las tarjetas de video, monitores de computadoras y otros dispositivos, y es casi universalmente llamado "HD-15" donde HD proviene de "alta definición" (*High Definition*), que lo distingue de los conectores que tienen el mismo factor de forma, pero sólo en 2 filas de pines.



Color	Red	Green	Blue	
Rojo	1111111111			
Verde		1111111111		
Azul			1111111111	
Amarillo	1111111111	1111111111		
Cian		1111111111	1111111111	
Magenta	1111111111		1111111111	
Blanco	1111111111	1111111111	1111111111	

PIN	Nombre	Descripción	Dir
Pin 1	RED	Entrada video color rojo	IN
Pin 2	GREEN	Entrada video color verde	IN
Pin 3	BLUE	Entrada video color azul	IN
Pin 4	N/C	Sin contacto	OUT
Pin 5	GND	Tierra (HSync)	.....
Pin 6	RED_RTN	Salida color rojo (GND)	.....
Pin 7	GREEN_RTN	Salida color verde (GND)	.....
Pin 8	BLUE_RTN	Salida color azul (GND)	.....
Pin 9	5V	5V (DC)	.....
Pin 10	GND	Tierra (VSync, DC)	.....
Pin 11	N/C	Sin contacto	OUT
Pin 12	SDA	I2C datos	OUT
Pin 13	HSync	Sincronismo horizontal	IN
Pin 14	VSync	Sincronismo vertical	IN
Pin 15	SCL	Reloj	OUT

Figura 2.22 Diagrama de conector VGA, tabla de pines y paleta de colores

2.6.2 Sincronización de señales SVGA.

Un solo píxel no lleva mucha información para formar una imagen. Una línea horizontal lleva un poco más. Pero un cuadro compuesto por múltiples líneas puede presentar una imagen en la pantalla del monitor. Un cuadro de video SVGA tiene típicamente 768 líneas, y cada línea contiene usualmente 1024 píxeles. Con el fin de colocar una imagen en la pantalla, existen circuitos de deflexión en el monitor que mueven los electrones emitidos por él o los cañones, de izquierda a derecha y de arriba abajo, a través de la pantalla del monitor.

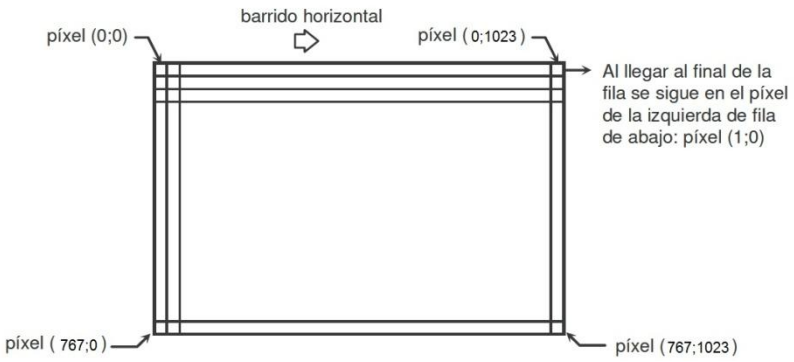


Figura 2.23 Barrido de la pantalla

Estos circuitos de deflexión requieren dos señales de sincronización para comenzar y terminar su funcionamiento en el momento indicado, de manera que se pueda representar

la línea de píxeles en la zona visible de la pantalla del monitor y ajustar una imagen desde arriba abajo. Los tiempos para la sincronización de señales SVGA dependen fuertemente de las especificaciones técnicas dadas por el fabricante del monitor.

En la siguiente tabla se pueden observar y comparar las diferentes frecuencias y señales de sincronismo tanto horizontal como vertical para algunas normas de video.

Formato	Reloj MHz	Horizontal (en píxeles)					Vertical (en líneas)				
		Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total	Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total
640x480@60Hz	25	640	16	96	48	800	480	9	2	29	520
640x480@60Hz	25,175	640	16	96	48	800	480	11	2	31	524
800x600@60Hz	40,000	800	40	128	88	1056	600	1	4	23	628
800x600@72Hz	50,000	800	56	120	64	1040	600	37	6	23	666
1024x768@60Hz	65,000	1024	24	136	160	1344	768	3	6	29	806
1024x768@75Hz	75,000	1024	24	136	144	1328	768	3	6	29	806

Figura 2.24 Tabla con distintas normas de video

Pulsos negativos en la señal de *sincronismo horizontal* marcan el comienzo y el final de una línea y aseguran que el monitor despliegue los píxeles entre los bordes, izquierdo y derecho, del área visible de la pantalla.

De una manera similar, pulsos negativos en una señal de *sincronismo vertical m*arcan el comienzo y el fin de una imagen, hecha de líneas de píxeles, y aseguran que el monitor despliegue las líneas entre los bordes superior e inferior de la pantalla del monitor.

2.6.3 Conversor D/A

El estándar SVGA es analógico y está diseñado para dispositivos de rayos catódicos (CRT). La fuente varía su tensión de salida con cada línea que emite para representar el brillo deseado. En una pantalla CRT, esto se usa para asignar al rayo la intensidad adecuada mientras éste se va desplazando por la pantalla. Este rayo no está presente en pantallas digitales (LCD); en su lugar hay una matriz de píxeles, y se debe asignar un valor de brillo a cada uno de ellos. El decodificador hace esta tarea tomando muestras del voltaje de entrada a intervalos regulares. Cuando la fuente es también digital (como un ordenador), esto puede provocar distorsión si las muestras no se toman en el centro de cada píxel, y, en general, el grado de ruido entre píxeles adyacentes es elevado.

El kit DE2 de Altera dispone del conversor A/D ADV7123. Como se ve en la Figura 2.25 permite trabajar con RGB en un formato de 10 bits para cada color; teniendo una variedad de 1024^3 colores diferentes.

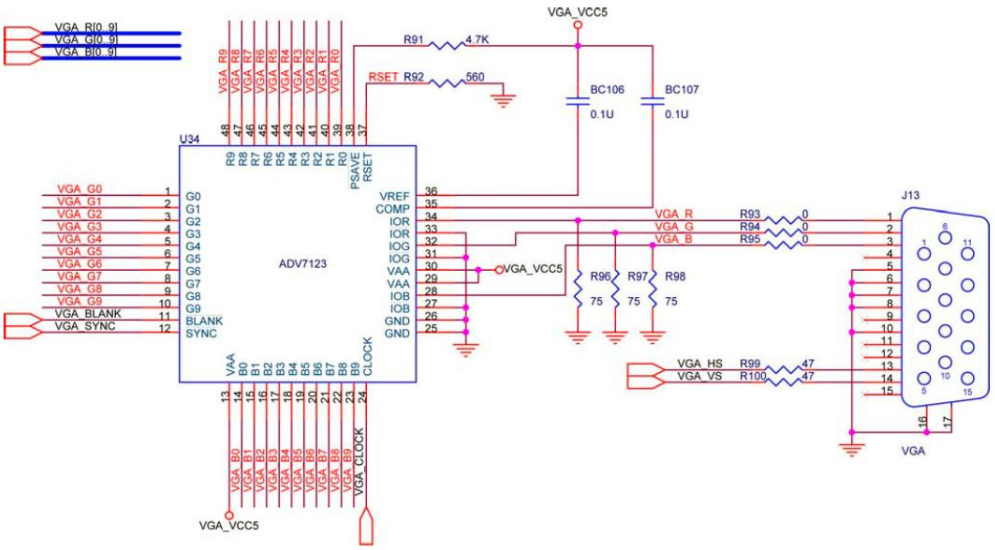


Figura 2.25 Representación RTL del conversor ADV7123 y la salida VGA del Kit DE2 Altera

Si se desea trabajar con una gama menor de colores para aplicaciones sencillas donde no haya que representar todos los matices de la imagen, basta con asignar los mismos valores a cada bit de cada uno de los colores red, green, blue. De esta forma se dispone de dos únicas posibilidades de palabra: *(111111111)b* o *(0000000000)b*. Esta elección limita la variedad de colores a 2^3=8 colores como se ve en la paleta de colores de la Figura 2.22.

Capítulo 3

Diseños y simulaciones preliminares

3.1 Primeras decisiones de diseño

En los primeros pasos de diseño, el análisis espectral parte de una señal cuya frecuencia máxima no supere los 10kHz; a través del cálculo de la FFT de 64 puntos. Este es el primer objetivo que se propone, con la idea de ocupar los menos recursos posibles del FPGA y acortar al máximo los tiempos de procesado, cálculo y dibujo de los resultados. Para este propósito se dispone del Kit de Desarrollo Nexys2 con un FPGA Spartan-3E, facilitado por el departamento de Telecomunicaciones y Electrónica de la Universidad de Pinar del Rio (Cuba). Asimismo para la realización de pruebas y simulaciones se hace uso del software Matlab R2011.

Analizando las propiedades del espectro con 64 puntos y un ancho de banda de 10kHz se comprueba que la resolución en frecuencia del espectro es excesivamente alta. Aplicando:

$$\Delta f = 1/NTs$$

siendo *N*=64=número de puntos de la FFT, *Ts=1/Fs* y cumpliendo el teorema de muestreo que impone *Fs*>2*20kHz; se obtiene una resolución de 156.25 Hz, algo excesiva.

Para resolver este problema se trabaja en el cálculo de un módulo de programación que calcule la frecuencia fundamental (*Fo*) de una señal periódica. El objetivo es conocida *Fo*, asignar *Fs=N*Fo*. De este modo a pesar de no aumentar la resolución del espectro, si que aumenta enormemente la calidad de los resultados obtenidos (Figura 3.1). Para ello se ha de calcular Fo con gran precisión.

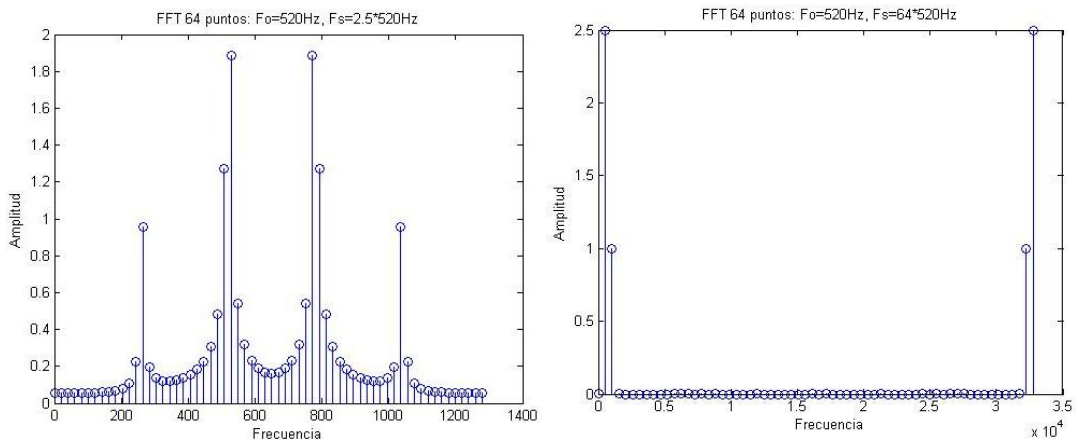


Figura 3.1: Simulación en Matlab de unaFFT 64 puntos con: Fs_1=2.5*520Hz , Fs_2=64*520Hz

Los esfuerzos se centran en la consulta de la bibliografía existente en determinación digital de la frecuencia fundamental de una señal; sin embargo, hay pocos algoritmos diseñados para tal fin y la implantación de un frecuencímetro digital se escapa de los objetivos del proyecto. Ateniendo al estudio de las propiedades de la FFT, se consigue diseñar un método iterativo que calcule el valor de *Fo* a través de los resultados de la FFT. Sin embargo, el número de puntos de la FFT es muy bajo y los tiempos aumentan a cada nueva iteración (en el AnexoI se muestra el código).

Por otro lado se realizan simulaciones de un algoritmo utilizando la función de autocorrelación, pero de nuevo aparecen dificultades; en este caso, la necesidad de una gran cantidad de muestras y tener que almacenarlas en una memoria, ocupa demasiado recursos del FPGA.

Ante esta situación y dado la limitada capacidad para acceder a búsquedas más generalizadas de información en la Universidad de Pinar del Rio, se decide muestrear a una frecuencia impuesta de 25kHz.

Una vez en la Universidad Pública de Navarra, con más recursos al alcance, se decide realizar ciertas mejoras en el dispositivo. Se dispone del Kit DE2 de Altera, con un FPGA Cyclone II EP2C35F672C, más potente que el anterior. En primera instancia, siguiendo la misma línea de acción se prueba con los métodos diseñados hasta el momento. Se aumenta el número de puntos de la FFT hasta 1024 y se aumenta el ancho de banda hasta 20kHz para abarcar el espectro completo de audio. Se hace una comparativa de las dos opciones planteadas hasta el momento. Con este fin se simulan las condiciones de trabajo con diferentes tipos de señales y de frecuencias de muestreo y en vista de que persisten los mismos defectos se plantea una nueva opción.

Analizando las posibilidades que ofrecen los analizadores de espectro comerciales. Se realiza un algoritmo que posibilite elegir al usuario diferentes opciones de span. El span, hace referencia al ancho de banda que se va a representar en la pantalla y equivale a realizar un zoom sobre un rango de frecuencias determinado.

Para lograr variar el ancho de banda, la señal debe pasar por un banco de filtros; a la salida se realiza un diezmado de los datos obtenidos y posteriormente se procesa la señal por el algoritmo de la FFT. Con este fin, se impone una frecuencia de muestreo fija que cumpla el teorema de Nyquist con $Fs>40KHz$ y se visualizan los resultados en el monitor. Mediante un botón el usuario puede realizar un zoom sobre frecuencias menores.

3.2 Diseño final

- Se fija un ancho de banda de 20kHz, permitiendo un análisis en frecuencia desde 2Hz hasta 20kHz, abarcando el rango de frecuencias audible por el ser humano.

- Como herramienta de cálculo se trabaja con una FFT de 1024 puntos.

- Se analiza le forma de representación de los datos en el monitor externo: aplicando la norma de video SVGA se dispone de 1024 píxeles en horizontal. Sin embargo el dibujo de dos componentes espectrales discretas separadas obliga a la representación de un máximo de 512 puntos. Esta limitación no lleva perdida de información; aprovechando la simetría de las componentes espectrales obtenidas por la FFT se selecciona únicamente el espectro positivo.

- El dibujo de los ejes, cuadrícula y texto de referencia, limita el espacio disponible en el monitor para dibujo de los 512 puntos. Con el objetivo de minimizar la pérdida de resultados se decide representar 491puntos.

- Con los datos anteriores se realizan diferentes simulaciones hasta llegar a la decisión de muestrear a 48kHz. Con este valor se aseguran los siguientes aspectos:

    1. Un ancho banda de la señal de 24kHz, que permite diseñar un filtro analógico paso bajo a dicha frecuencia minimizando la distorsión producida de la señal para frecuencias menores o iguales a 20kHz.

    2. Por otro lado permite el dibujo en el monitor del espectro hasta la frecuencia máxima de 20kHz impuesta en el diseño

- Se posibilita la opción de elegir entre 4 opciones de span. De tal forma que sea viable hacer un zoom espectral sobre componentes de frecuencia más bajas. Aplicando los mismos criterios que en el punto anterior se obtiene las frecuencias máximas representadas en el monitor en cada span. *Fs,v* representa la frecuencia a la que se transmite cada nueva muestra dependiendo del span.

Opción	Span	*Fs,v*
0	20kHz	48kHz
1	12kHz	24kHz
2	4kHZ	8kHz
3	1kHz	2kHz

La elección de las frecuencias de corte máxima representadas en cada opción de span se ha basado en la documentación acerca del análisis de audio y de las simulaciones realizadas. Como referencia señalar que la línea telefónica puede transmitir frecuencia de hasta 4kHz, realizando el muestreo a 8kHz y los CD de audio captan señales de hasta 20kHz con una tasa de muestreo de 44.1kHz

En la Figura 3.2 se muestra una simulación del diseño realizado. En este caso se trabaja con una señal con alto contenido armónico para exponer de una forma clara el objetivo y las ventajas que suponen variar el span.

En primera instancia se muestrea a un frecuencia de 48kHz y se realiza la FFT de 1024 puntos. Debido a la existencia de multitud de componentes armónicas, el usuario no puede hacer una distinción clara de donde se sitúa cada una de ellas, ya que muchas quedan enmascaradas en un intervalo demasiado próximo. Al variar el ancho de banda, se filtran las frecuencias más altas; en el caso de un span de 12kHz, esta será la frecuencia de corte. Si se realiza de nuevo el cálculo de la FFT la resolución disminuye casi el doble y se hace una mejor distinción del espectro. Este proceso se repite para cada una de las opciones de span hasta llegar a una frecuencia de corte de 1kHz.

Este proceso de variación de span se lleva a cabo mediante una técnica denominada diezmado. De tal forma que el muestreo de la señal se realiza a una frecuencia fija de 48kHz. Cuando se filtra la señal se impone una frecuencia de corte *fc=Fs/2C*, donde *C* represente el factor de reducción o diezmado. De esta manera para conseguir que la salida se simule a una frecuencia mayor a 2*fc* se selecciona una muestra de cada C que salen del filtro. Por tanto a la salida del módulo diezmador se tendrá las muestras tomadas a una frecuencia de 2*fc*.

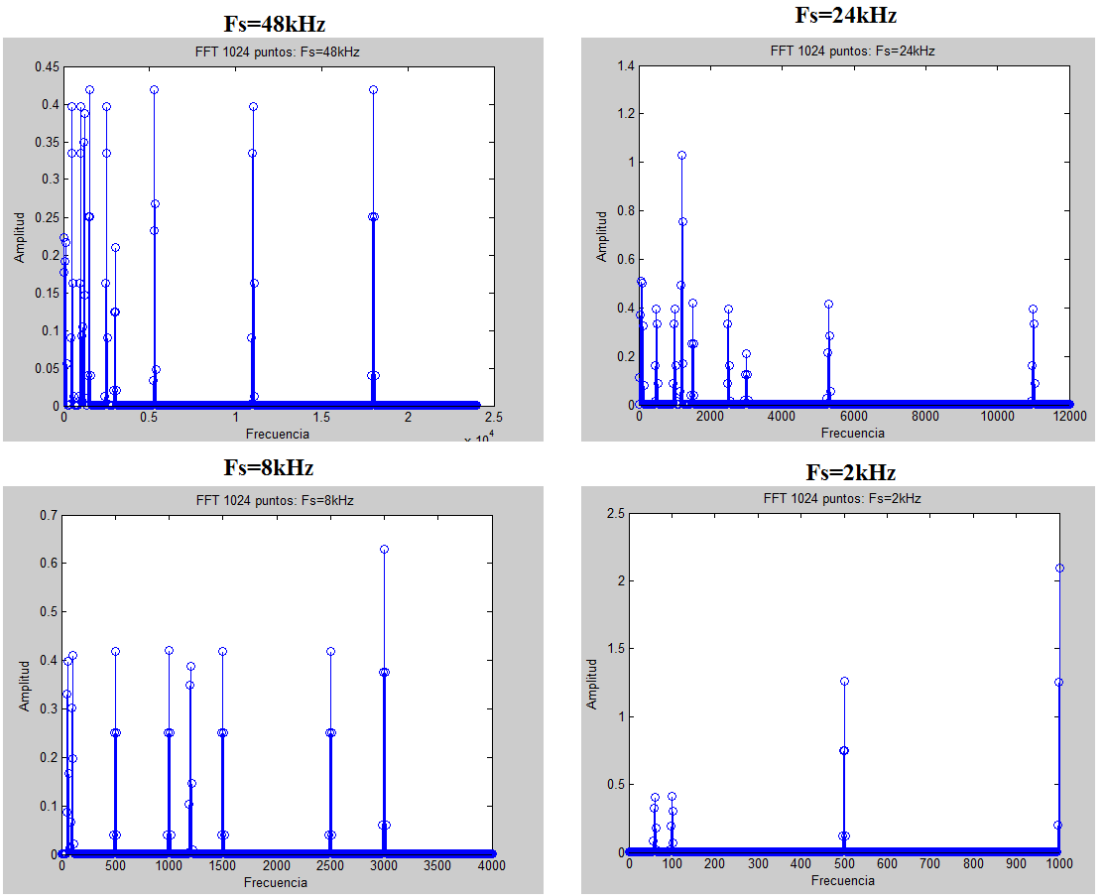


Figura 3.2: FFT 1024 puntos con: Fs_1=48kHz Fs_2=24kHz, Fs_3=8kHZ, Fs_4=2kHz

El siguiente paso en el diseño resuelve dos cuestiones inherentes a las criterios establecidos: la existencia de aliasing en el muestreo de una señal con *Fo<Fmax* y la optimización del método de enventanado. Hasta el momento se ha elegido una ventana rectangular. Para ello, a continuación se hace una introducción a cada uno de los dos aspectos.

3.3 Aliasing y filtros digitales

Al muestrear una señal cuya frecuencia máxima es mayor que la frecuencia de muestreo, las componentes de frecuencia superiores a *Fs* son reconstruidas y aparentarán ser frecuencias por debajo de la frecuencia de muestreo, fenómeno conocido como aliasing.

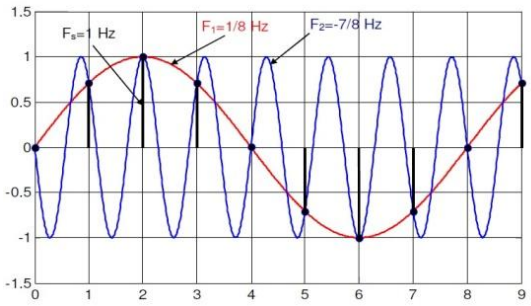


Figura 3.3: Aliasing

Para evitar esta situación (Figura 3.3) se debe realizar un filtrado paso bajo de la señal. Dadas que en el diseño se tiene cuatro frecuencia máximas a analizar, se hace necesario implantar 4 filtros pasa bajo para cada caso. No obstante, el diseño de un circuito analógico con 4 filtros se complica y aumenta el número de componentes utilizados. Para resolver esto se decide el siguiente diseño:

- Filtro analógico paso bajo con frecuencia de corte (fc) en 24kHz.
- Tres filtros digitales para cada una de las 3 opciones de span con *Fs,v* inferiores a 48kHz. Las frecuencias de corte de cada uno se fijan en *Fs,v*/2.

A la hora de elegir el tipo de filtro digital a diseñar se hace una comparativa entre los existentes: destacando los filtros FIR e IIR

3.3.1Filtros FIR e IIR

Los filtros digitales lineales e invariantes en el tiempo pueden clasificarse de acuerdo a la longitud de su respuesta impulsiva *h[n]* como IIR, cuando la respuesta tiene duración infinita o FIR, si su duración es finita. Esta respuesta impulsiva *h[n]*, *n = 0,1,2,...* caracteriza completamente el filtro, a punto tal que las señales de entrada y salida están relacionadas por la suma de convolución, que para filtros IIR toma la forma

$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k] \qquad (1)$$

Y para filtros FIR

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \qquad (2)$$

Del análisis de estas dos ecuaciones es evidente que, mientras que la suma convolución puede ser una forma apropiada para implementar un filtro FIR, no es adecuada para los filtros IIR debido a que la respuesta impulsiva es muy larga (en teoría, infinitamente larga). Por ello, los filtros IIR se implementan con ecuaciones a diferencia que permiten calcular las muestras de salida en forma recursiva

$$y[n] = -\sum_{k=1}^{N} a_k x[n-k] + \sum_{k=0}^{M} b_k x[n-k] \qquad (3)$$

donde $n = 0, 1, 2, \ldots, y[-1] = 0$. El número N es el orden del filtro, y fija la cantidad de modos de la respuesta impulsiva. La relación entre los coeficientes a_i y b_i se obtiene aplicando la transformada Z y antitransformando.

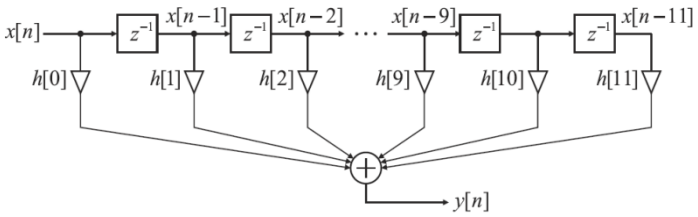


Figura 3.4: Estructura filtro FIR donde orden 11

3.3.2 Comparación entre filtros FIR e IIR

La elección entre una implementación FIR e IIR depende de las ventajas relativas de cada uno de estos dos tipos de filtros.

1. Los filtros FIR se pueden diseñar para tener una respuesta de fase estrictamente lineal (distorsión de fase nula), lo que es importante en muchas aplicaciones, como transmisión de datos, audio digital y procesamiento de imágenes. La respuesta de fase de filtros IIR no es lineal, en especial en cercanías de la zona de transición.
2. Los filtros FIR implementados de forma no recursiva, por ejemplo aplicando *(2)*, son inherentemente estables. En cambio, la estabilidad de los filtros IIR siempre debe comprobarse, ya que son sistemas realimentados.
3. Los efectos causados por la implementación con aritmética de punto fijo, tales como los errores de cuantización de los coeficientes y los errores por redondeo en las operaciones aritméticas, son mucho más severos en los filtros IIR que en los FIR.
4. Para satisfacer unas especificaciones dadas los filtros FIR necesitan un mayor número de coeficientes que los filtros IIR, sobre todo si las bandas de transición son estrechas. En consecuencia, los requerimientos de memoria, el número de operaciones y los tiempos de procesamiento son mayores para los FIR que para los IIR. Sin embargo, la posibilidad de implementar los FIR mediante la técnica de convolución rápida usando

FFT y también el empleo de técnicas *multirate* permite aumentar significativamente la eficiencia de las implementaciones.

5. Un filtro analógico convencional puede convertirse en un filtro digital IIR equivalente que satisfaga las especificaciones de diseño de manera sencilla. Esto no es posible con filtros FIR pues no tienen una contraparte analógica. Sin embargo es más sencillo sintetizar filtros con respuestas en frecuencia arbitrarias utilizando filtros FIR.

De las características detalladas arriba puede esbozarse una guía tentativa para elegir entre una implementación FIR o IIR:

- Si los únicos requerimientos importantes son bandas de transición estrechas (filtros con cortes muy abruptos) y eficiencia de cómputo, se prefieren filtros IIR pues necesitan un número de coeficientes mucho menor que un filtro FIR equivalente.
- Si el número de coeficientes del filtro no es muy elevado (por ejemplo, si las bandas de transición no son muy abruptas), y en particular, si se desea muy poca o ninguna distorsión de fase, se suele elegir filtros FIR. Los procesadores digitales modernos (DSP) están optimizados para implementar este tipo de filtros, y algunos se han diseñado específicamente con esa finalidad.

Debido a estos aspectos detallados se decide implementar un tres filtros FIR.

3.4 Enventanado

El enventanado de señal es una técnica de procesado que consiste en aplicar una función limitada en un intervalo temporal *[0, T]*, multiplicándola sobre la señal de interés que se quiere estudiar. De este modo, se extrae un fragmento de señal para realizar sobre él análisis espectral, dejando a cero todas las muestras que queden fuera del intervalo mencionado. El objetivo de aplicar una ventana es observar el espectro de un segmento dado de la señal, de un número finito de muestras, de modo que se observen sus características para ese trozo concreto, y aplicando sucesivas ventanas que vayan extrayendo fragmentos consecutivos, apreciar la evolución del espectro en función del tiempo. El intervalo de señal sobre el que se aplica la ventana es interesante que sea .

Los parámetros fundamentales que determinan el enventanado de una señal son su longitud temporal, es decir, el tamaño del intervalo extraído, que será del número de puntos de la FFT y la función que define la ventana. De esta manera la acción de enventanar una señal se define como la multiplicación en el tiempo de la señal por la función ventana:

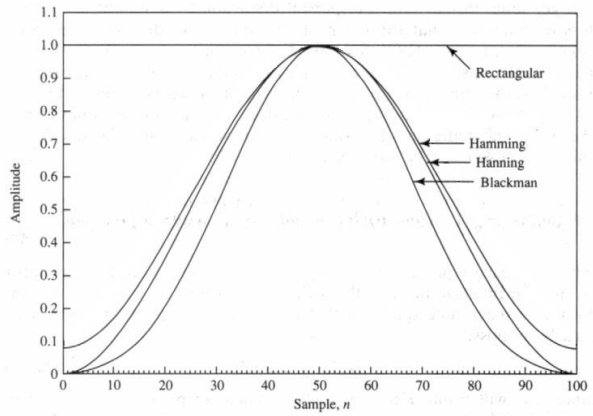
$$Sw(t)=s(t)w(t)$$

por tanto, en el espectro, el efecto será realizar una convolución entre las transformadas de la señal y la ventana:

$$Sw(f)=s(f)*w(f)$$

La elección de la función ventana determinará como influirán los transitorios que se dan en los extremos del segmento de la señal enventanada sobre el resultado final. Lo que interesa es buscar una función ventana que modifique lo menos posible el espectro de la señal de interés al producirse la convolución de ambas. Lo deseable, como se puede deducir, sería utilizar la función delta. La convolución de cualquier señal sobre una delta es la misma señal, intacta. Sin embargo, esto es irrealizable en la práctica, pues supondría no afectar a la señal en el tiempo, es decir, no realizar el enventanado realmente.

Para abordar este problema, existe una familia de funciones ventana que permiten realizar una aproximación. Actúan seleccionando las muestras del intervalo *[0, T]*, tal y como hace la ventana uniforme, pero además, suavizan el valor de las muestras al comienzo y al final de la misma. De este modo, en el espectro, esto supone convolucionar la transformada de la señal original por el de la ventana, que consistirá en un lóbulo central y una serie de lóbulos secundarios anexos, lo que en realidad es una aproximación a la función delta.



Resolución espectral	
Ventana	Resolución
Rectangular	$2\pi/L$
Hamming	$4\pi/L$
Hanning	$4\pi/L$
Blackman	$9\pi/L$

Figura 3.5: a) Gráfica ventanas superpuestas b) Resolución espectral

Se pueden citar como algunas de las ventanas más conocidas la Hanning, la Hamming, Blackman-Harris, Bartlett, etc. Sin entrar en detalles sobre las características particulares de cada una, es necesario mencionar que no existe ninguna ventana que pueda considerarse la más óptima en términos absolutos, todas ellas implican tomar una decisión de compromiso entre la resolución frecuencial y temporal que se desea obtener, lo cual, depende del tamaño de los lóbulos principal y secundarios de la transformada de cada una de ellas.

3.4.1 Ventanas usadas en el proyecto

Las ventanas que se habilitan para el uso en el sistema son:

- Ventana Rectangular
- Ventana Hanning
- Ventana Hamming
- Ventana Blackman

La importancia de cada una de las ventanas, radica en las características de inicio y de fin de las ventanas, las cuales permiten disminuir los efectos de las discontinuidades al momento de realizar el enventanado de una señal.

Ventana	Ancho del lóbulo principal	Nivel de lóbulo principal a secundario (dB)
Rectangular	$4\pi/M$	13
Hanning	$8\pi/M$	27
Hamming	$8\pi/M$	43
Blackman	$12\pi/M$	58

Figura 3.6 Tabla características de las ventanas utilizadas

A la hora de elegir la ventana, lo ideal sería la función delta que tiene dos características principales: separación de lóbulos secundarios infinita y anchura de banda nula.

Ante la imposibilidad se debe llegar a un compromiso teniendo en cuenta que una separación de lóbulos laterales finita supone la aparición de un cierto rizado (espectro lobulado). Una anchura de banda no nula supone que las transiciones en frecuencia sean menos abruptas (suavizado espectral). Los lóbulos secundarios enmascaran las componentes de poca amplitud.

Por otro lado, se debe considerar también el efecto de ponderación temporal, puesto que con las ventanas tipo coseno alzado, las muestras de los extremos de la ventana quedan minimizadas frente a las muestras de la zona central de la ventana.

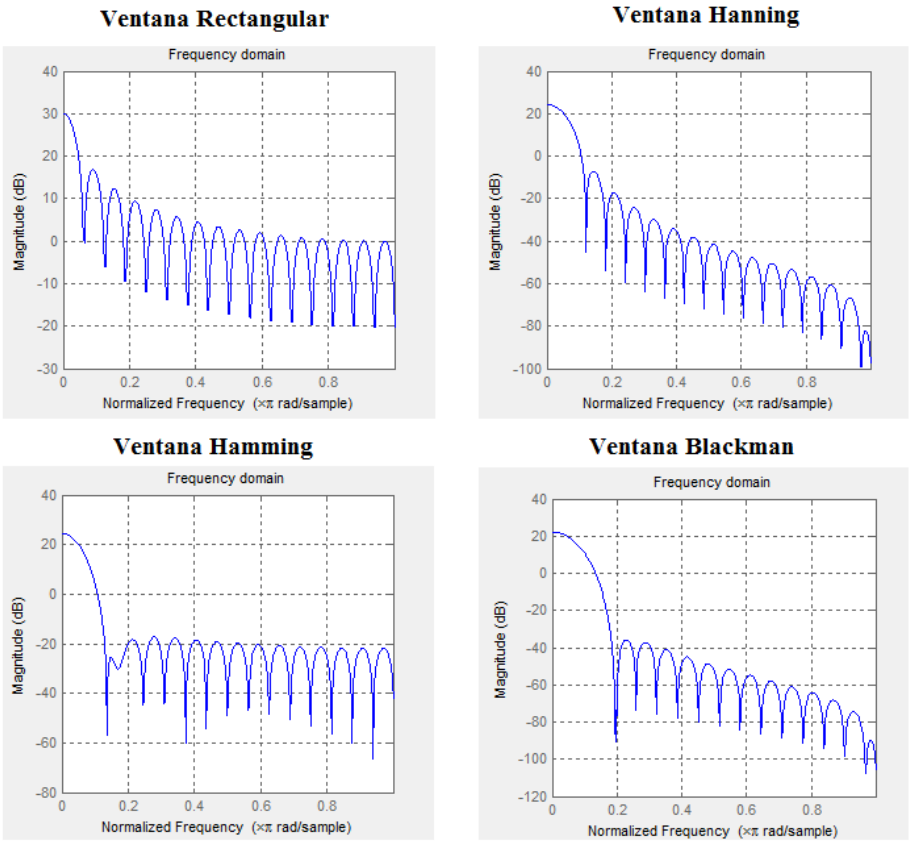


Figura 3.7 Cuadro comparativo del diagrama de Boode de cada ventana

Este cuadro comparativo junto a la tabla de las Figuras 3.6 y 3.7 muestran de forma clara la eficiencia de cada una de las ventanas. En primera instancia se ve que todas las gráficas presentan errores en cuanto a la relación de las amplitudes. También se puede observar que las amplitudes alcanzan una magnitud similar. En cuanto a otros parámetros como el ancho de banda del lóbulo principal, se determina que las ventanas de Hamming y Rectangular tienen un lóbulo principal muy definido, pero la atenuación de las frecuencias parásitas no es tan eficiente como con otras ventanas. Por lo que se obtiene las siguientes conclusiones:

- La ventana Blackman va a proporcionar un mayor suavizado espectral y un buen comportamiento minimizando el rizado. En contraposición la ventana Rectangular presenta el mayor rizado y el menor suavizado del espectro.

- Respecto la ventana Hamming y la Hanning van a tener un comportamiento bastante parecido. Estos resultados se deducen también de la Figura 3.5b), dado que a mayor resolución mayor es la dispersión.

Capítulo 4

Desarrollo del sistema e implementación

4.1 INTRODUCCIÓN

4.1.1 Primeros pasos

Los primeros pasos en la programación del analizador de espectro sobre FPGA fueron un vuelco en la revisión de la bibliografía que versa sobre el tema [1]. De esta forma, se descubrieron diversos artículos donde se proponían diseños parecidos y se definía la estructura general del sistema. Una vez definida dicha estructura, la forma de actuar dio paso a la realización de diagramas de bloques esquematizando los módulos principales y sus funciones. Un punto importante en el diseño han sido las limitaciones impuestas por el hardware disponible: principalmente por el conversor analógico-digital y por el FPGA.

En el trabajo desarrollado en la Universidad Hermanos Saíz Montes de Oca de Pinar del Rio (Cuba) se disponía de un FPGA XCS500E de la familia Spartan 3E empotrado en el Kit de Desarrollo Nexys2 junto al conversor A/D PmodAD1 de la compañía Digilent.

Posteriormente en la Universidad Pública Navarra los resultados se adaptaron para el kit DE2 de Altera operando con un FPGA EP2C45F672C6ES y al conversor MAX164 de la casa Maxim.

4.1.2 Estructura general analizador de espectros

La estructura general definida del sistema se compone de tres bloques principales:

- Etapa de acondicionamiento de la señal
- Desarrollo del software en el FPGA para implementar los algoritmos de cálculo de las componentes espectrales
- Visualización en un monitor externo (RTC o LCD) según la norma.

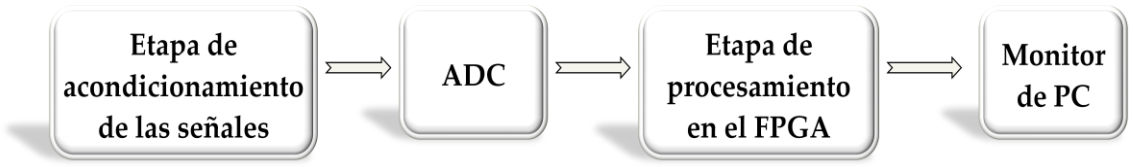


Figura 4.1 Diagrama de bloques de la estructura general

A continuación, en las siguientes secciones se describe con detalle cada uno de los bloques que conforman el sistema

4.2 ETAPA DE ACONDICIONAMIENTO DE LA SEÑAL

El diseño de esta etapa corresponde a un circuito analógico cuya finalidad es la adecuación de la señal de entrada conforme a las características del conversor y del análisis en frecuencia que se va a realizar. Las dos limitaciones que van a condicionar dicho diseño son:

- la tensión máxima de entrada de la señal al conversor A/D.
- la frecuencia máxima a la que se realiza el muestreo.

En el marco teórico se decide el desarrollo de una analizador de espectro para cualquier señal periódica de entrada en un rango de tensión (-5,5)V y de frecuencia entre (0,20)kHz. Para el desarrollo del proyecto se dispone del conversor A/D MAX163 de la casa Maxim. Se trata de un conversor paralelo cuyas características principales son las siguientes:

- Resolución: 12 bits
- Tiempo de conversión: 8.33 us → Frecuencia conversión máxima: 120kHz
- Voltaje de entrada: (0,5)V.

Como se puede observar la entrada solo permite una señal unipolar máxima de 5V. Esta circunstancia hace indispensable el uso de un mecanismo que nos convierta la señal de bipolar a unipolar. Adicionalmente, se debe asegurar una protección del sistema que nos restringa la tensión máxima de la señal para evitar destruir el montaje en caso de error en el uso del dispositivo.

Por otro lado, el conversor no impone ninguna restricción respecto a la frecuencia de conversión. En el diseño experimental del proyecto, mediante simulaciones en otros software como Matlab, se ha decidido imponer una frecuencia máxima de muestreo de 48kHz; por tanto la limitación parte de que se requiere un filtro pasa bajo que atenúe las componentes de frecuencia superiores a la frecuencia de corte *fc*, con el fin de evitar el fenómeno de aliasing. Igualmente a través de la experimentación y la simulación, se decide diseñar un filtro con *fc* de 24kHz.

Por último para lograr que el sistema tenga una impedancia de entrada muy elevada y una impedancia de salida casi nula, es necesario conectarle a la entrada del circuito un amplificador operacional en configuración seguidor de voltaje (buffer). De esta manera, se aíslan las dos partes del circuito, haciendo que el segundo no resulte una carga para el primero, pues la impedancia vista será muy alta.

Con estos cuatro puntos se tiene ya clara la finalidad de esta etapa y los pasos siguientes que se deben dar. A continuación se desarrollan las decisiones y cálculos realizados para llevar a la práctica los anteriores puntos.

4.2.1 Protección, buffer y sumador no inversor

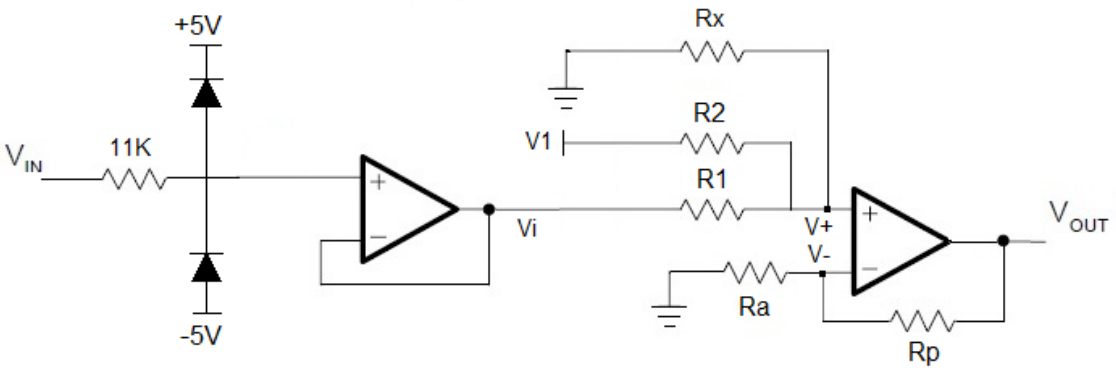


Figura 4.2 Esquema electrónico sistema protección, buffer y sumador

Los dos diodos en esta disposición nos aseguran que si la tensión a la entrada supera V_{cc}^{+}+0.7 o V_{cc}^{-}-0.7; el valor de la señal queda anclado a dichas tensiones, siendo 0.7V la tensión umbral de los diodos. Para el resto de voltaje de entrada, la señal no se ve afectada, siendo V_{IN}. Se toman como valores $V_{cc}^{+} = 5V$ y $V_{cc}^{-} = -5V$.

El siguiente componente en el esquema es un amplificador con realimentación negativa en modo seguidor de tensión; en esta configuración la salida ve la misma tensión que la entrada, siendo $V_i = V_{IN}$

Por último se introduce un amplificador no inversor sumador, cuya ecuación característica es: $V_{out} = Vi/2 + V1/2$ con $V1$=5V (1)

Atendiendo a *(1)*, para la conversión de bipolar a unipolar se escala por 2 la tensión de entrada y se le suma una componente de directa u *offset* de valor 2.5V. El uso de un amplificador no inversor tiene la ventaja de que no introduce un desfase de 180º como la configuración inversora; conservando al máximo la calidad de la señal. Para la elección del valor de los componentes, acorde con la ecuación *(1)* deseada para el modelado de su comportamiento, se realizan los siguientes cálculos. Haciendo un análisis del circuito se tiene que:

$$V_{out} = (1 + \frac{Rp}{Ra})V^{-} = \frac{Rp}{(Rp \parallel Ra)}V^{-} \qquad (2)$$

$$V^{+} = \frac{(R1 \parallel R2)^2 Rx}{(R1 \parallel R2) + Rx}(\frac{Vi}{R1} + \frac{V1}{R2}) \qquad (3)$$

Dado que $V^{+} = V^{-}$, sustituyendo en las ecuaciones *(2)* y *(3)*, nos queda:

$$V_{out} = \frac{Rp(R1 \parallel R2)^2 Rx}{(Rp \parallel Ra) + Rx}(\frac{Vi}{R1} + \frac{V2}{R2}) \qquad (4)$$

Para la elección de las resistencias se han escogido valores del orden de kΩ, de tal modo que $R1=R2=2$kΩ y $Rp=Ra=Rx=1$kΩ. Sustituyendo en *(4)* se resuelve la ecuación característica deseada *(1)*.

Las siguientes gráficas muestran las simulaciones llevadas a cabo mediante el software *Proteus* para validar el correcto funcionamiento del circuito antes de su implementación física.

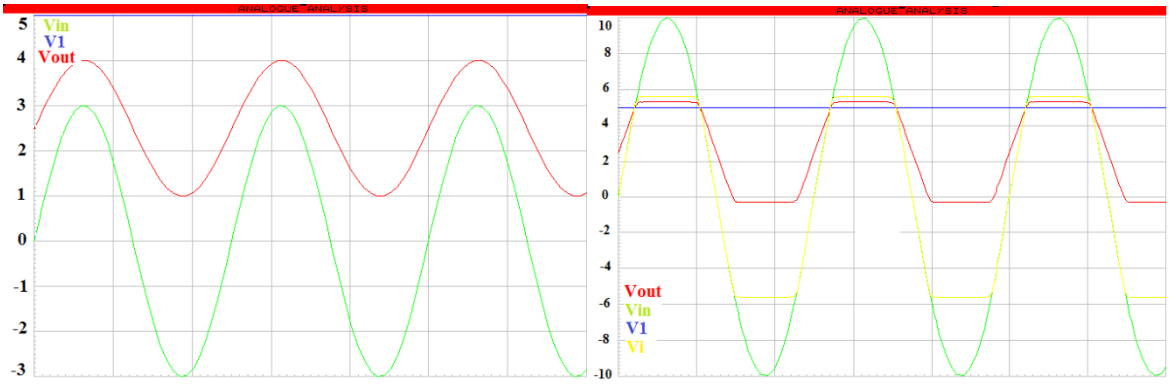


Figura 4.3 Simulaciones realizadas con el software Proteus del circuito de protección, buffer y sumador a) V_{IN}>5V b(V_{IN}<5V

4.2.2 Filtro pasivo paso bajo Butterworth de tercer orden

Como se ha comentado anteriormente, otro punto importante en el diseño de la etapa de acondicionamiento es la introducción de algún tipo de filtro para evitar el fenómeno de alyasing. Para este fin, especificado una frecuencia máxima de muestreo de 48kHz y una frecuencia máxima de la señal a analizar de 20kHz se decide diseñar un filtro paso bajo con una frecuencia de corte de 24kHz. Esta frecuencia se sitúa por encima de la frecuencia máxima de análisis para evitar la atenuación o distorsión de la señal en la máxima medida posible; en especial, en señales cuadradas, donde el efecto del filtro es más evidente.

En base a la experimentación y a la bibliografía específica (Figura 4.4) se simulan diferentes tipos de filtros variando las características de entrada para finalmente escoger un filtro pasivo paso bajo Butterworth de tercer orden bajo una configuración conocida como Sallen-Key.

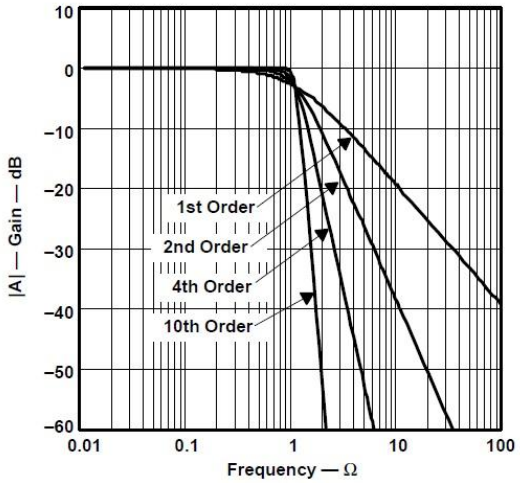


Figura 4.4 Respuesta de la ganancia de los filtros Butterworth de diferentes orden respecto la frecuencia normalizada *Ω(Ω=f/fc)*

La expresión general para esta respuesta es:

$$|H| = \frac{1}{\sqrt{1+(w/w_c)^{2n}}}$$

Donde n es el orden del filtro y w_c la frecuencia de corte, la cual en este caso representa la magnitud de -3dB. Como se puede apreciar en la Figura 4.4 la respuesta es extremadamente plana cerca de la componente de corriente directa, algo redonda cerca de la frecuencia de corte y la respuesta decae a un ritmo de -20n dB/dec en la banda de atenuación,

El filtro de paso bajo Butterworth maximiza la planitud de la respuesta de magnitud dentro de la banda de paso. Por lo tanto, son de uso frecuente en aplicaciones de conversión de datos, donde se requieren niveles exactos de las señales a través de toda la banda de paso. La Figura 4.5 muestra la respuesta de la magnitud frente a la frecuencia del filtro diseñado.

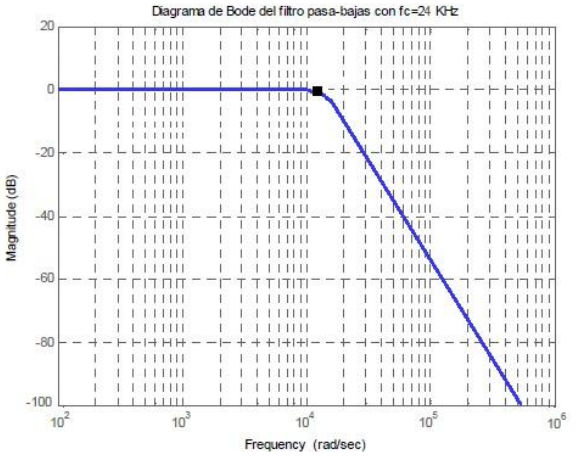


Figura 4.5 Diagrama de Boode del filtro diseñado

A continuación se detalla el diseño de un filtro Butterworth pasa bajo con ganancia unitaria en una configuración *Sallen-Key*. Para obtener una mejor respuesta en la señal filtrada se optó por implementar un filtro Sallen-Key de tercer orden. Para lograr este cometido basta con añadir un filtro de primer orden en serie con un filtro de segundo orden [FA.1].

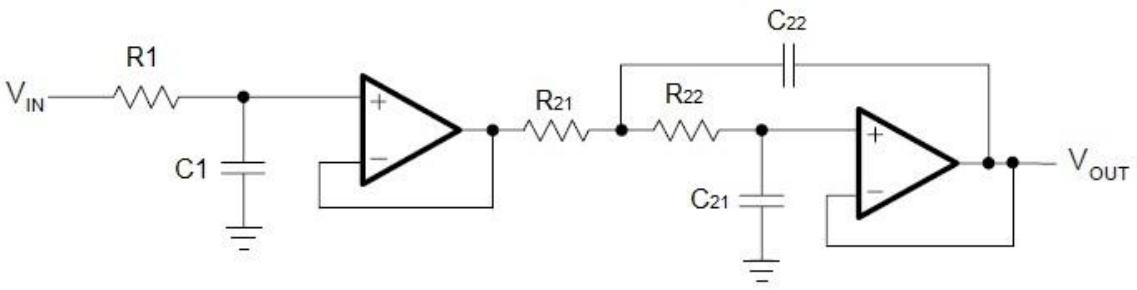


Figura 4.6 Esquema electrónico del filtro diseñado: el primer amplificador modela un filtro de primer orden; el segundo uno de segundo orden

La función de transferencia de la ecuación *(1)* representa un filtro de segundo orden; mientras que la ecuación *(2)* representa un filtro de primer orden.

$$A_2(s) = \frac{Ao}{(1 + a_2 s + b_2 s^2)} \quad (1) \qquad A(s) = \frac{Ao}{(1 + a_1 s)} \quad (2)$$

Para el filtro de orden 2 de la Figura 4.6 se tiene que la función de transferencia es:

$$A_2(s) = \frac{Ao}{1 + w_c[C_{21}(R_{21} + R_{22}) + (1 - Ao)R_{21}C_{22}]s + w_c^2 R_{21}R_{22}C_{21}C_{22}s^2} \quad (3)$$

Dada una ganancia unitaria *(Ao=1)*, simplificando la ecuación:

$$A_2(s) = \frac{1}{1 + w_c C_{21}(R_{21} + R_{22})s + w_c^2 R_{21}R_{22}C_{21}C_{22}s^2} \quad (4)$$

La igualdad entre la ecuación *(4)* y *(1)* da como resultado:

$$Ao = 1 \quad (5)$$

$$a_2 = w_c C_{21}(R_{21} + R_{22}) \quad (6)$$

$$b_2 = w_c^2 R_{21}R_{22}C_{21}C_{22} \quad (7)$$

Despejando R_{21} y R_{22} en función de C_{21} y C_{22} se tiene:

$$R_{21,12} = \frac{a_2 C_{22} \pm \sqrt{a_2^2 C_{22}^2 - 4b_2 C_{21}C_{22}}}{4\pi f_c C_{21}C_{22}} \quad (8)$$

En orden a satisfacer valores reales para el resultado de la raíz cuadrada se debe cumplir que:

$$C_{22} \geq C_{21}\frac{4b_1}{a_2^2} \quad (9)$$

Análogamente para el filtro de primer orden comparando el circuito de la Figura 4.6 con la ecuación *(2)* se tiene que:

$$R1 = \frac{a_1}{2\pi f_c C1} \quad (10)$$

Para el cálculo de los coeficientes *ai* y *bi* se hacer uso de unas tablas de coeficientes normalizadas como la de la Figura 4.7:

n	i	a_i	b_i	$k_i = f_{Ci} / f_C$	Q_i
1	1	1.0000	0.0000	1.000	—
2	1	1.4142	1.0000	1.000	0.71
3	1	1.0000	0.0000	1.000	—
	2	1.0000	1.0000	1.272	1.00

Figura 4.7 Tabla coeficientes diseño filtro Butterworth

Una vez agrupadas todas las ecuaciones con los valores de los coeficientes se pasa a calcular los valores numéricos de las resistencias y condensadores. Empezando con el filtro de primer orden, dado que se tiene una ecuación y dos incógnitas se arranca con el valor de partida de *R1*=10kΩ y aplicando *(10)* se resuelve que *C1*=680uF.

Para el filtro de segundo orden se presenta la misma dificultad: 5 ecuaciones, 6 incógnitas. En esta caso, se parte de un valor dado a C_{21}=330pF. Aplicando (9) se tiene $C_{22} \geq 1.32$nF. Debido a escoger un valor normalizado se impone C_{22}=3.3nF. Resolviendo la ecuación *(8)* se completan los resultados con: R_{21}=2.2kΩ y R_{22}=18kΩ.

El esquema con el valor de todos sus componentes se muestra en la Figura 4.8

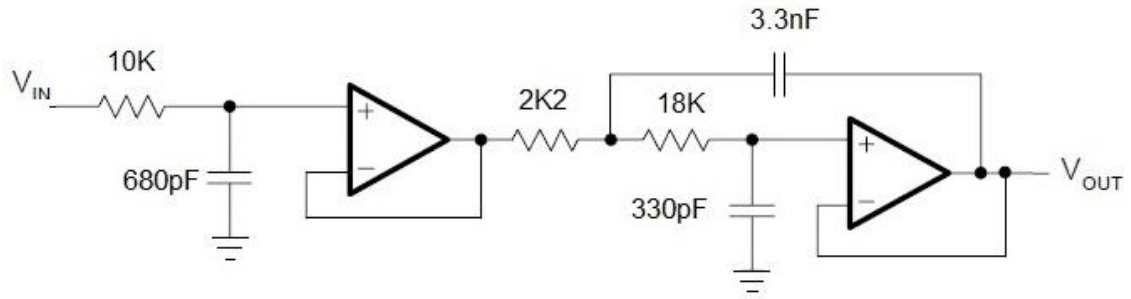


Figura 4.8 Esquema final filtro Butterworth de 3º orden bajo configuración Sallen-Key

Por último, en la Figura 4.9 se muestra el esquema completo del circuito introducido, con el valor de sus componentes,

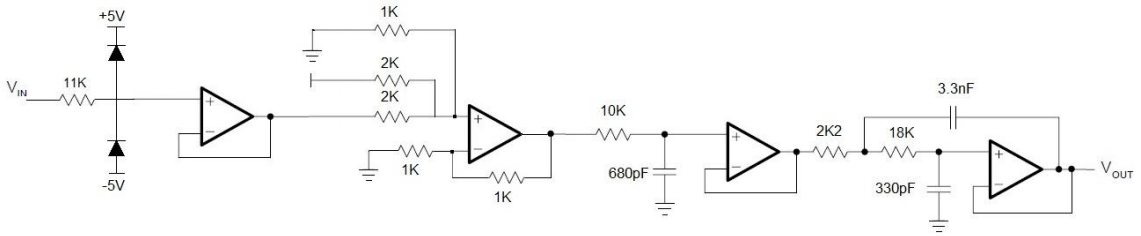


Figura 4.9 Esquema final etapa de acondicionamiento de la señal

En la etapa de diseño y posteriormente de implantación física y conexionado de circuito electrónico se hace uso del amplificador operacional TL084CN. Las ventajas que ofrece son: un buen comportamiento en las condiciones de frecuencia en las que se trabaja y acoplamiento de 4 amplificadores operacionales en un mismo encapsulado; de tal forma que minimiza el uso de recursos y el espacio en la placa.

4.2.3 Conversor A/D

El conversor es el encargado de la digitalización de las señales a procesar por el FPGA. Es un interfaz entre el mundo analógico y el digital; la señal de entrada continua se discretiza en un tren de pulsos de tensión. Como se ha comentado anteriormente para el desarrollo del proyecto se trabaja con el conversor A/D MAX163. Algunas propiedades técnicas importantes se especifican a continuación:

Resolución =12 bits
Error de offset = ±4 LSB*
Voltaje de señal de entrada = (0,5)V
Voltaje entradas lógicas

- Entradas lógicas *RD* y *CLKIN* <=’0’ Vmax= 0.8V
- Entradas lógicas *RD* y *CLKIN* <=’1’ Vmin= 2.4V
- Entradas lógicas *D11-D0/8* y *BUSY* <=’0’ Vmax= 0.4V
- Entradas lógicas *D11-D0/8* y *BUSY* <=’1’ Vmin= 4V

El error de offset es la diferencia entre el valor obtenido realmente con entrada 0V y el valor ideal binario; en esta caso *(000000000000)b*. Se expresa en LSB* (*last significant bit*); de tal forma que si Vref=-5V; $(-5V/2^{12})*(\pm 4)= \pm 4.88mV$=La tensión que hay que aplicar a la entrada para tener un valor nulo en la salida.

A continuación se describen cada una de las entradas y salidas del conversor (Figura 4.10):

AIN:	Entrada analógica a muestrear	
Vref:	Tensión de referencia	= -5V
AGND:	Tierra analógica	
V_{SS}:	Alimentación negativa	= -15V o -12V
V_{DD}:	Alimentación positiva	= 5V

<u>Entradas lógicas de control</u>

HBEN:	Cuando está en alto sirve para multiplexar la conversión interna de 12 bits en órdenes menores. Para nuestro objetivo permanecerá en bajo; por tanto se conecta a tierra.
CLKIN:	Reloj de entrada para sincronismo de señales
RD/*CS*:	Señales de control. CS se conecta a tierra por razones que se detallaran más adelante.

<u>Salida lógicas de control</u>

BUSY:	Indica que hay una conversión en proceso mediante un valor en bajo

<u>Salidas lógicas</u>

D3/11-D0/8:	Salidas de datos
D11-D4:	Salidas de datos
CLKOUT:	Reloj de salida. Se deja abierto.

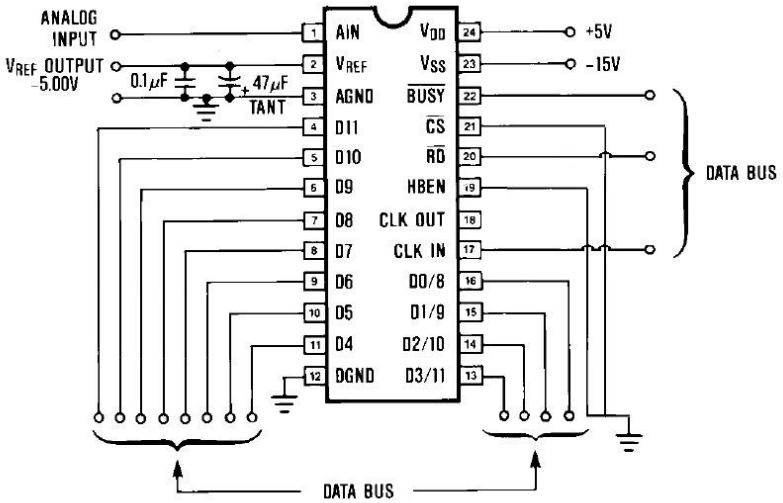


Figura 4.10 Esquema encapsulado conversor A/D MAX163

Para controlar el funcionamiento del conversor se diseña un algoritmo en VHDL que controla las señales correspondiente a: "orden inicio conversión" y un reloj interno de sincronismo que requiere el conversor. Estas dos señales corresponden a las pines 20 y 17, *RD* y *CLKIN* respectivamente. La última de las señales que se requiere es BUSY, salida del conversor, que manda la información de "convirtiendo".

Todas estas señales se conectan a través de una cinta como la de la figura para poder ser conectadas/desconectadas a su vez a los pines de expansión de entrada del FPGA de una forma rápida, fácil y fiable que cause la mínima introducción de ruido.

Figura 4.11 a) Cinta para conexión bus de datos en b) el cabeza de expansión del FPGA

Adicionalmente hacer referencia a que todas las tierras del circuito son comunes, incluida la que sale del FPGA, para de este modo evitar "tierras virtuales" a distinto voltaje.

4.2.4 Control de conversor A/D

El conversor MAX163 posibilita la elección de diversas opciones de control. Analizadas cada una de ellas se opta por: *Slow Memory Mode.* El diagrama de control se muestra en la Figura 4.12. La señal *CS* valida *RD* cuando su nivel lógico es bajo '0'; para mayor simplicidad se ancla permanentemente a dicho valor conectando a tierra.

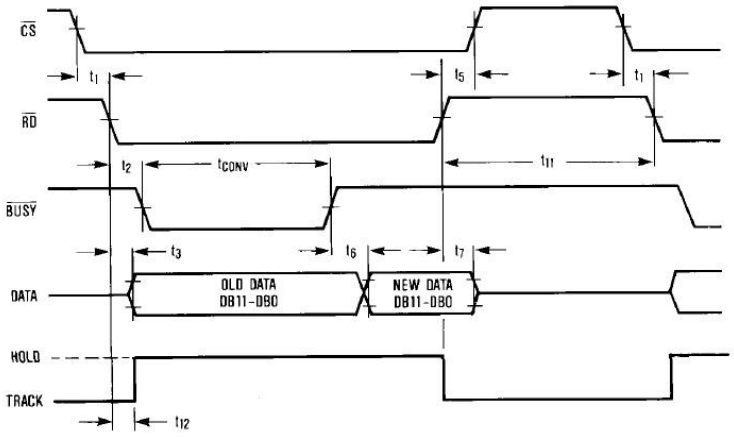


Figura 4.12 Control del conversor A/D

RD marca la orden de comienzo de conversión cuando esta a nivel lógico bajo. En nuestro diseño se modela como un reloj a la frecuencia de muestreo elegida, con un ciclo útil del 20%. Esto se debe a que el valor del tiempo de conversión (t_{conv}) es en el peor de los casos 13 ciclos de *CLKIN* (1MHz); es decir 13us. Para una frecuencia de muestreo máxima de 48kHz se tiene que el tiempo en bajo de *RD* es de 17 us, asegurando de este modo una completa conversión.

Por último la señal BUSY nos sirve como indicador de que se introduce al sistema una nueva muestra tomando el instante en que pasa de su valor lógico alto a bajo (*falling_edge*).

La Figura 4.13 muestra la implementación física realizada en el laboratorio de todo el circuito analógico; así como de la conexión al FPGA. Adicionalmente una vez validado el funcionamiento se diseña una placa PCB para minimizar la entrada de ruido y distorsiones.

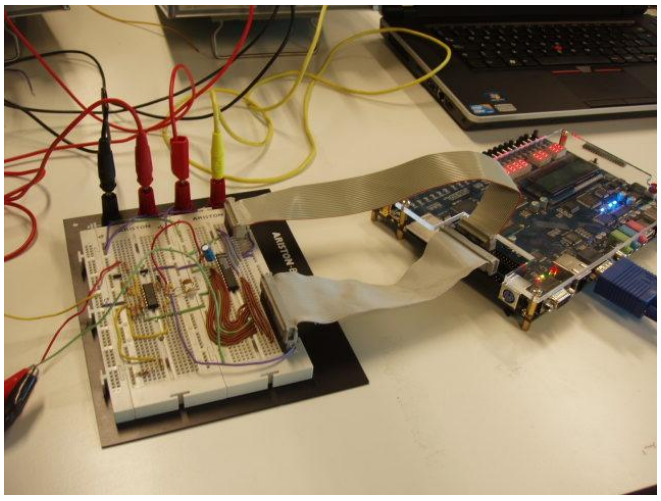


Figura 4.13 Instantánea circuito montado

4.3. ETAPA DE PROCESAMIENTO EN FPGA

4.3.1 Programación en VHDL

El propósito de un diseño en VHDL para síntesis con lógica programable es obtener un circuito lógico sobre un dispositivo de lógica programable a partir de una descripción VHDL. Para ello, se sigue un proceso determinado por un flujo de diseño, donde se indica cada fase del desarrollo del circuito digital.

Con VHDL se puede definir el funcionamiento de un circuito lógico textualmente con sentencias que definen su interfaz y comportamiento. La descripción del circuito se realiza principalmente con expresiones lógicas, componentes y procesos de señales; sin embargo el lenguaje permite abstraer las funciones lógicas y codificar en alto nivel los algoritmos de funcionamiento del circuito.

Para la implementación del diseño de un sistema en un FPGA, el diseñador cuenta con la ayuda de entornos de desarrollo especializados. Altera proporciona varias herramientas de desarrollo como son: *SOPC Builder* (herramienta que permite al usuario crear un sistema basado en el procesador Nios II) y *Quartus II*, entre otras, siendo esta última la seleccionada en este proyecto.

El sistema de desarrollo *Quartus II* es una plataforma de herramientas para el diseño de circuitos digitales sobre dispositivos FPGA y CPLD de Altera. Quartus II provee aplicaciones para la entrada de diseño, síntesis lógica, simulación lógica, ubicación y conexionado, análisis temporal, administración de potencia y programación de dispositivos, junto con una variedad de utilitarios y aplicaciones adicionales para el diseño lógico programable.

A su vez en el proyecto se hace uso del software *Modelsim* que administra un entorno que permite editar, compilar, simular y depurar diseños de sistemas digitales descritos en VHDL a través de Quartus II.

4.3.2 *IP-Cores*

Para facilitar la programación de funciones complejas, en el diseño se implantan módulos *IP-Cores* (Núcleos de Propiedad Intelectual). Los *IP-Cores* son núcleos de programación que incluyen la funcionalidad de varios circuitos integrados. Pueden ser desarrollados por el mismo equipo de trabajo o adquiridos a través de un tercero. Debido a que estas funcionalidades son como componentes electrónicos, pero sin su parte física, se los suele llamar componentes virtuales. En la industria se los conoce como bloques de propiedad intelectual o *IP*-Cores

Idealmente, un *IP*-Core debe ser lo más portable posible, es decir, que fácilmente se pueda adaptar a cualquier tecnología de otros proveedores o a diferentes métodos de diseño.

El proceso de diseño a alto nivel ha generado nuevos mercados de *IPs* orientados a la venta de celdas complejas ya diseñadas y verificadas, que pueden ser incorporadas dentro de la jerarquía de diseño como celdas de librería *OpenCores*; iniciativa para la creación de una extensa biblioteca de IPs.

Altera les nombre como *MegaCores*, los cuales han sido verificados y testeados en los FPGAs de Altera minimizando los tiempos de diseño y verificación. Como ventaja adicional son completamente parametrizables, permitiendo: una rápida y fácil visión de la documentación, especificación de los parámetros, configuración de herramientas de trabajo de terceros y generación de los ficheros necesarios para la integración del *IP* de Altera dentro del diseño.

4.3.3 Metodología de diseño y estructura general

La metodología de diseño llevada a cabo en el proyecto es *Top-Down* y consiste en dividir el sistema en varios bloques de manera que se puedan resolver los problemas por separado. Cada bloque a su vez se puede dividir en otros si es necesario. El objetivo es que cada bloque tenga una función específica representada como un componente con entradas y salidas que desempeñan dicha función.

En base a ello el primer paso que se da es aunar todas y cada una de las especificaciones deseadas de una manera detallada, con la ayuda de esquemas y diagramas de bloques.

La estructura general de la programación realizada se muestra en la Figura 4.14. El esquema básico consta de tres modulo conectados entre sí y con el exterior por señales de entrada y salida. Cada módulo por sí mismo es independiente y permite la simulación para comprobar su correcto funcionamiento; así como una depuración posterior del código sin grandes esfuerzos.

Figura 4.14 Diagrama de bloques estructura de la programación

P1: Controlador del Conversor: como su nombre anuncia controla el correcto funcionamiento del conversor A/D. Este modulo enlaza con el siguiente mediante una señal de 12 bits que transmite los datos de adquisición.

P2: Analizador: realiza diferentes funciones destacando como principal el cálculo de la FFT, realizado mediante una función *MegaCore*.

P3: Dibujo SVGA: es el interfaz de salida con el monitor externo; mediante un controlador y un conversor D/A permite visualizar los resultados.

A continuación se describen los objetivos que se buscan en la programación:

- Análisis espectral de una señal con componente de frecuencia máxima de 20kHz.
- Posibilidad de zoom espectral variando factor de diezmado a través de un botón del Kit DE2.
- Posibilidad de elección de 4 opciones de enventanado a través un segundo botón.
- Posibilidad de visualización del espectro con el eje de magnitud en escala lineal (V) y logarítmica (dB) a través de un interruptor
- Posibilidad de reseteo del sistema a través de un segundo interruptor

Definidos los objetivos y la estructura general de la programación, se analiza el módulo Top del diseño. Cada una de las entradas y salidas que tiene se conectan con el exterior a través del conector VGA, del kit DE2 (relojes, botones, interruptores) y de la banda de bus de datos de la etapa de acondicionamiento. A continuación se muestra el diagrama (Figura 4.15) y se detallan las salidas y entradas.

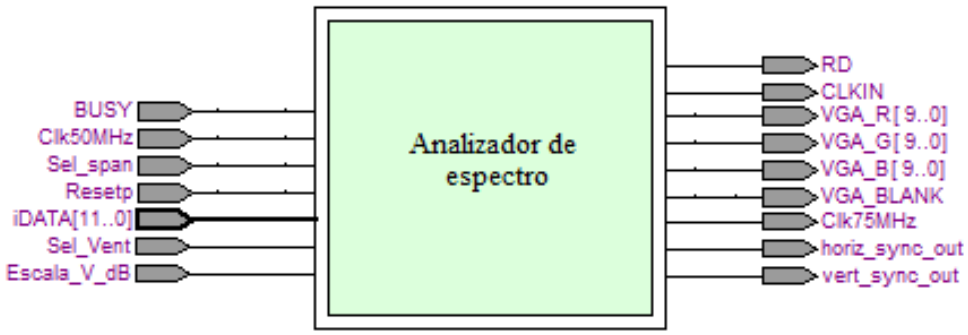


Figura 4.15 Módulo principal con sus entradas y salidas

Las entradas:

Entradas que vienen del conversor:

BUSY	Señal de control del conversor A/D
iDATA	Señal con los datos muestreados (vector de 12 bits)

Entrada del propio FPGA

Clk50MHz	Señal del reloj maestro del sistema
Sel_span	Señal elección de span a través del boton1 [KEY0]
Sel_vent	Señal elección enventanado a través botón 2 [KEY1]
Resetp	Señal de reseteo del sistema a través interruptor 1 [SW0]
Escala_*V_dB*	Señal escala lineal o logarítmica a través interruptor 2 [SW1]

Las salidas:

Salidas al convertidor D/A

VGA_R[0:9],VGA_G[0:9], VGA_B[0:9], VGA_BLANK, clk75MHz

Salidas al cable VGA

Horiz_sync_out, vert_sync_out

Salida al conversor

RD,CLKIN Señales de control del conversor A/D

Cada una de estas señales son asignadas a un pin del FPGA a través de un interfaz que proporciona el software de Quartus II. Las conexiones se detallan en el Capítulo 5.

A continuación se hace una descripción de cada uno de los bloques señalados en el diagrama de la Figura 4.14 analizando el objetivo y el comportamiento que presentan, los conocimientos necesarios para su diseño y las partes del código más representativas.

4.3.4 Controlador conversor

El siguiente módulo es el encargado de controlar el funcionamiento del conversor a través de dos señales: *CLKIN* y *RD* y de obtener las muestras discretas que sirven de entrada al siguiente módulo para el cálculo de la FFT, junto a la señal de control y sincronismo: *BUSY*.

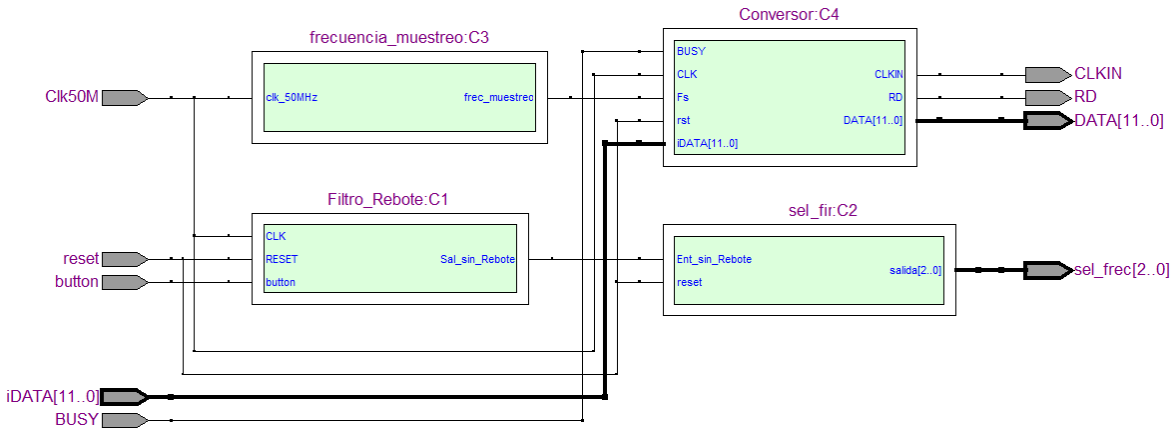


Figura 4.16 Esquema de los módulos del: Controlador Conversor

Consta de cuatro módulos conectados entre si, dos de ellos para generar las señales de control de span a partir de la elección del usuario, y los restantes para el control de conversor. De esta manera se tiene:

C1: Filtro_rebote: Es un filtro diseñado para eliminar las componentes de mayor frecuencia generadas por el rebote que se produce al presionar el botón 1 [KEY0], de tal manera que se obtenga un único pulso a su salida (*Sal_sin_rebote*).

C2: sel_fir: Se encarga de contabilizar el número de pulsos que le entran mediante un contador de 3 bits. Una vez el contador toma el valor 4 se inicializa a 0. La salida *sel_frec* sirve para seleccionar el filtro diezmador encargado de procesar las muestras que entran.

De este modo se tiene:

- Inicio span 20kHz *sel_frec =”000”*
- 1º pulso span 12 kHz *sel_frec =”001”*
- 2º pulso span 4 kHz *sel_frec =”010”*
- 3º pulso span 1 kHz *sel_frec =”011”*
- 4º pulso inicio *sel_frec =”000”*

C3: frecuencia_muestreo: Es un divisor de frecuencia respecto el reloj del sistema a 50MHz. Mediante un proceso cuenta los pulsos del reloj *clk_50MHz* a través de un contador. Cuando este es igual a la señal *cant_pulsos* se inicializa a 0 y sigue el proceso. La salida *frec_muestreo* es la señal de control de conversión del conversor y corresponde a un reloj de 1 bit a la frecuencia de muestreo con un ciclo útil del 20%.

```
architecture Behavioral of frecuencia_muestreo  is

--cant_pulsos=50.000.000/48.000
signal cant_pulsos: STD_LOGIC_vector (14 downto 0) := " 000010000010001" ;
signal contador: STD_LOGIC_vector (14 downto 0) := "000000000000000" ;
signal altop: STD_LOGIC_vector (40 downto 0);
signal alto: STD_LOGIC_vector (14 downto 0);
begin

process (clk_50MHz)
begin

        if clk_50MHz'event and clk_50MHz = '1' then
                if contador = cant_pulsos then
                        contador <= "000000000000000";
                        else
                                contador <= contador + 1 ;
                        end if ;
end if ;
end process ;

altop   <=      cant_pulsos*"0110011001100110011001100";
alto    <=      altop(39 downto 25);

frec_muestreo <=        '0' when contador <= alto else
                                                '1';
end Behavioral;
```

Figura 4.17 Código de programación del módulo C3: frecuencia_muestreo

C4: Conversor: Controla el conversor a través de tres procesos:

- Divisor de frecuencia de 50MHz a 1MHz: Nos da la señal *CLKIN*
- Señal de control *RD*: Cuando *reset* esta a nivel lógico bajo (desactivado) a *RD* se le asigna la señal frecuencia de muestreo. Cuando *reset* esta activado *RD* vale ‘1’.

- Adquisición de datos: La señal proveniente del conversor *BUSY* cuando sufre un cambio lógico de alto a bajo marca la introducción de un nuevo dato. El dato (señal *DATA*) se codifica en un vector (std_logic_vector) de longitud 12 bits.

4.3.5 Analizador

Este modulo es el corazón del analizador de espectro, en el se realizan las operaciones descritas posteriormente (Figura 4.18) para lograr los siguientes objetivos:

- Conversión de unipolar a bipolar.
- Banco de filtros FIR para filtrar y diezmar las muestras cuando el usuario elija una menor opción de span.
- Cuatro opciones diferentes de enventanado.
- Algoritmo FFT.
- Adecuación de los resultados para posterior procesado por el módulo de dibujo SVGA y escritura de los mismos en una memoria RAM.

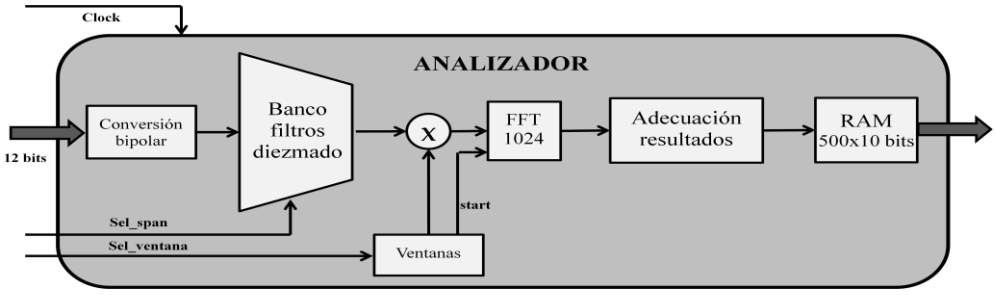


Figura 4.18 Diagrama de bloques del módulo P2: Analizador

En primer lugar se analizan las señales de entrada:

DataIn_Rest: recibe los datos del conversor a una resolución de 12 bits.
DataRdy: es un reloj a la frecuencia de muestreo sincronizado con la señal *BUSY* que sale del conversor. De esta manera se dispone de una señal que a cada flanco de bajada de un pulso indicando el comienzo de un nuevo dato.
*cont*_out: Contador de 8 bits para el control de la memoria RAM.
Reset: Señal de reset del sistema.
V_dB, vent, selec_frec: Son las señales elegidas por el usuario para variar la escala, la ventana utilizada y el span.

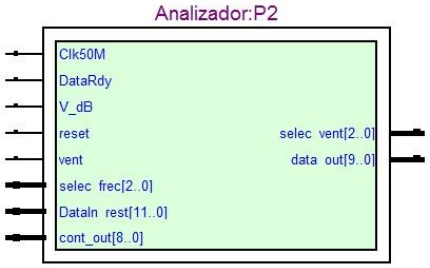


Figura 4.19 Módulo P2: Analizador

A continuación se describen cada uno de los bloques que componen el P2:Analizador.

A1:	*conv_bi*	Módulo conversión de unipolar a bipolar
A2:	*Banco_FIR*	Banco de filtros FIR
A3:	*envent*	Módulo de enventanado de la señal
A4:	*FIFO*	Memoria FIFO
A5:	*General_FFT*	Módulo general de cálculo de la FFT
A6:	*General_mod*	Módulo general de cálculo del módulo de los resultados
A7:	*General equiv*	Módulo general de cálculo equivalencias para SVGA
A8:	*Memoria*	Memoria RAM
A9:	*enables*	Módulo generación reseteos y habilitadores
A10:	*Filtro_rebote*	Filtro eliminación rebote de las pulsaciones del botón

4.3.5.1 Conversión de unipolar a bipolar

El conversor utilizado se limita a valores de tensión positivos entre 0 y 5V; dado que las señales de entrada han pasado en la etapa analógica por un amplificador sumador que las convierte en unipolares, se requiere de la etapa inversa.

Para la reconversión simplemente el módulo *A1: conv_bi* resta el valor de continua sumado anteriormente. El factor a restar (*R*) se calcula teniendo en cuenta la resolución del conversor, al ser unipolar $R=(2^{12}/5)*2.5=2^{11}$. Al efectuar la resta y como se trabaja en un formato de punto fijo se requiere de un bit para el signo; por tanto no es necesario multiplicar por 2 como se hizo en la etapa analógica.

La entrada de datos a este módulo está controlada por una señal de reset que proviene de un interruptor del kit. Una vez habilitada, el flujo de datos de entrada al módulo P2: Analizador es continuo.

4.3.5.2 Banco de filtros FIR

Recibe los datos muestreados de la señal ya convertidos a bipolar. Este módulo A2: *Banco_FIR* está controlado por la selección de span hecha por el usuario. Cada uno de los filtros lleva una señal de enable que se activa cuando se presiona el botón del kit [KEY0].

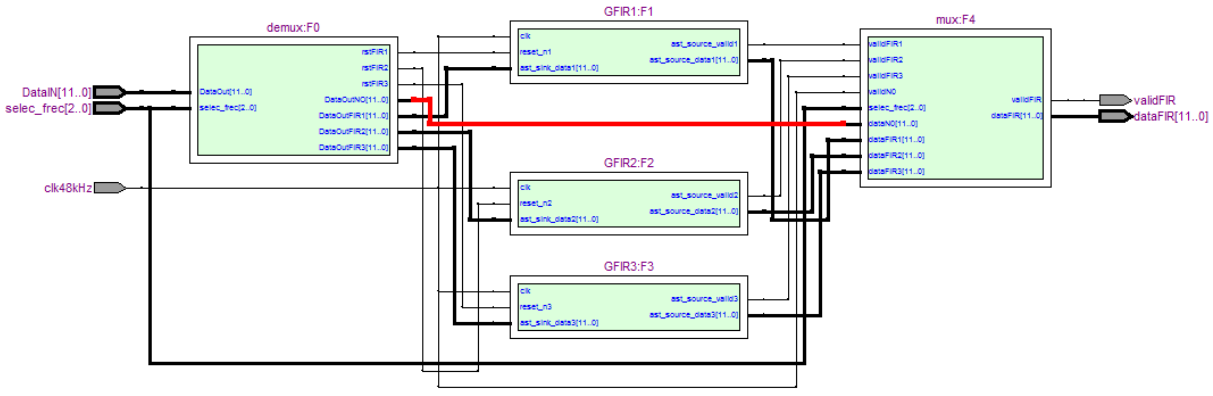


Figura 4.20 Esquema del módulo A2: Banco_FIR

Los filtros usados corresponden a filtros FIR diezmadores, diseñados con el método de las ventanas. Para su implantación se ha hecho uso de un *MegaCore.* Los tres utilizan una ventana Hanning de 100 coeficientes y utilizan una estructura de aritmética distribuida en paralelo. La configuración utilizada para una frecuencia de muestreo de 48kHz es:

	Frecuencia de corte	Factor diezmado
FIR1 :	12kHz	2
FIR2 :	4kHz	6
FIR3:	1kHz	24

Si el usuario no presiona el botón o presiona una cuarta vez, la señal no pasa por ninguno de los filtros realizándose el análisis del espectro completo de la señal con una frecuencia máxima de 20kHz (señal roja Figura 4.20)

4.3.5.3 Enventando

A continuación las muestras pasan al módulo A3: *envent*, en referencia a enventanado. La función es multiplicar 1024 muestras por los coeficientes de la ventana elegida por el usuario. El proceso de elección de la ventana es análogo al de elección de la frecuencia de muestreo. Se ha implantado la opción de elegir entre cuatro ventanas, siendo estas: Rectangular, Hamming, Hanning y Blackman. El orden de selección corresponde al orden anteriormente expuesto.

La precisión con la que se trabaja a la hora de definir los coeficientes de enventanado es de 12 bits, 1 para la parte entera y 11 para la parte decimal, trabajando con una resolución de 2^{-11}=0.000488281. Una vez realizada la multiplicación se selecciona solo la parte entera del resultado.

Este módulo tienes dos salidas. Por un lado la señal *data_out* transmite los datos enventanados; y por otro, la señal *pulso,* de 1 bit, indica con un pulso en alto la salida de cada nuevo dato.

4.3.5.4 FIFO

El objetivo de implantar una memoria FIFO (*First input First output*) radica en que los datos de entrada a la FFT deben estar sincronizados con un reloj más rápido que la frecuencia de muestreo. Por ello los datos son leídos a cada pulso del reloj principal del sistema a 50MHz.

Las dos entradas principales del módulo A4: *FIFO* son las muestras y la señal *pulso* que sale del módulo A3: *envent* indicando cada nuevo dato. Para la programación no se hace uso de un *MegaCore*; sino que los datos se escriben y leen en un array bidimensional compuesto por 1024 vectores de 12 bits cada uno. Las señales de control inicio/fin escritura/lectura son controladas por otro módulo. De esta forma automáticamente las 1024

muestras terminan de ser escritas, se activa una bandera que permite la lectura e impide la escritura hasta nueva orden. A la salida del modulo se tiene tres señales que nos permiten el control del siguiente: un contador de 11 bits, una señal que indica a nivel lógico alto la lectura del paquete de datos la señal de control que habilita la lectura de la FIFO.

4.3.5.5 FFT

Para el cálculo de la FFT se hace uso de un *MegaCore* facilitado por la librería de Altera. Mediante una pantalla se permite la configuración de las distintas opciones para implantar el algoritmo adecuado a las especificaciones de diseño. De esta forma se eligen las siguientes:

Longitud de transformación:	1024 puntos
Precisión datos:	12 bits
Factor twiddle:	12 bits
I/O Data Flow :	Variable Sreaming
Orden de entrada/salida:	Natural
Representación datos:	Punto Fijo

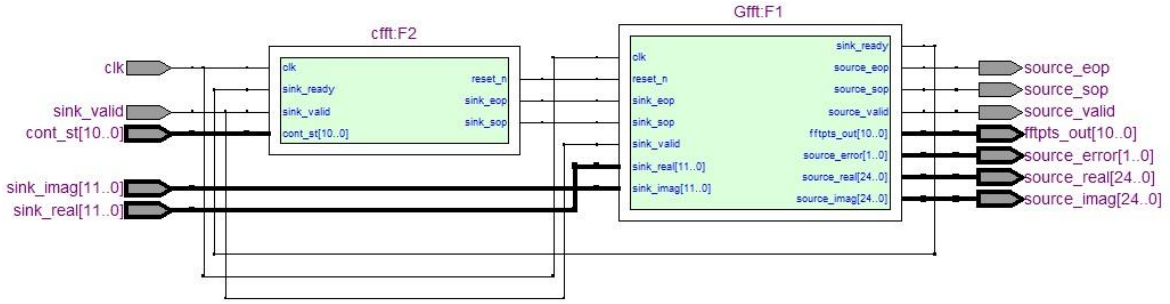


Figura 4.21 Estructura módulo General_FFT

La Figura 4.21 muestra las señales de entrada y salida y las respectivas conexiones del *MegaCore* de la FFT (F1: Gfft) y del módulo de control de las señales (F2: cfft). Adicionalmente se representa una simulación del funcionamiento del *Megacore* y del algoritmo de control para facilitar la comprensión de cada una de las señales.

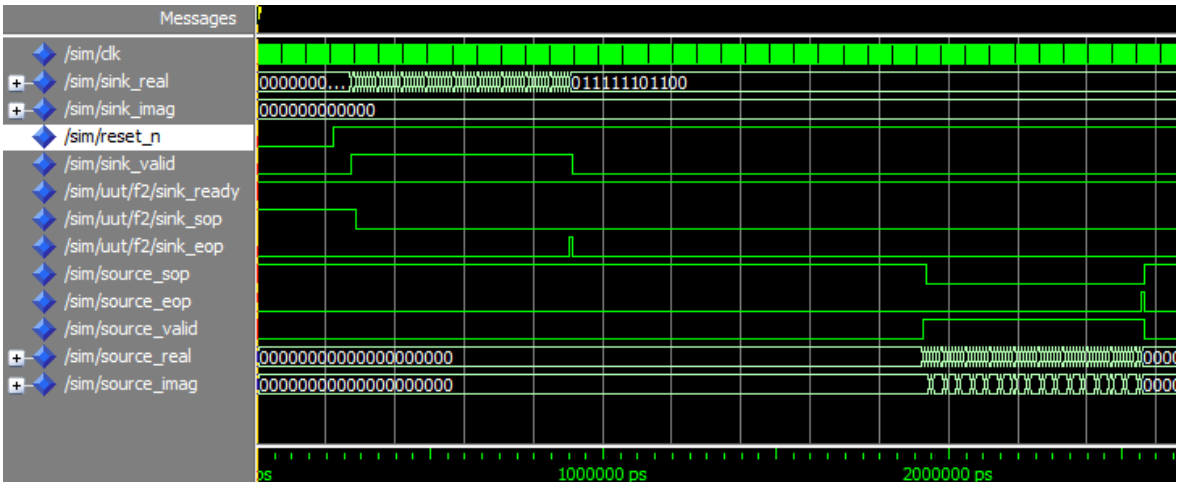


Figura 4.22 Simulación realizada con la herramienta *ModelSim*

La ejecución del proceso para cálculo de la FFT está controlado por sus entradas, por ello se hace un análisis de cada una; obviando las entradas y salidas de las muestras reales e imaginarias (*sink_real, sink_img, source _real, source_img*), en nuestro caso sink_imag se deja permanentemente a '0'.

- *Sink_valid*: Indica si los datos recibidos son válidos
- *Sink_sop*: Indica el comienzo de un paquete de datos.
- *Sink_eop*: Indica el fin de un paquete de datos
- *Reset_n:* Se activa en alto siempre 3 pulsos de reloj antes de recibir datos habilitando la FFT; una vez termina el proceso vuelve a nivel bajo. Por otro lado a *Sink_valid* se le asigna la señal de salida de la FIFO habilitando la entrada del paquete de datos.

Solo quedan los pulsos de comienzo y fin de un paquete de datos. Para su control se crea un contador que cuente el número de muestras que entran en cada paquete (1024) con la condición que *sink_valid* y *sink_ready* estén en alto. Sink_ready es una señal de salida que indica si la FFT está lista para recibir datos. El resto de señales siguientes son de salida:

- Source_ready: Indica si el resto de módulos está listo para recibir los resultados
- Source_sop: Indica el comienzo de un paquete o símbolo de resultados
- Source_eop: Indica el fin de un paquete o símbolo de resultados.
- Source_valid: Indica si los resultados son válidos '1' o no '0'

```
signal cnt: std_logic_vector (10 downto 0):="00000000000";
begin
reset_p: process( cont_st)
 begin
        if cont_st="00000000010" then
                reset_n<='0';
        elsif cont_st="01111111101" then
                reset_n<='1';
        end if;
end process reset_p;
cnt_p : process (clk, sink_valid)
 begin
        if rising_edge(clk) then
                if  sink_ready = '1' and sink_valid='1' then
                        if cnt = "01111111111" then
                                cnt <= "00000000000";
                        else
                                cnt <= cnt + '1';
                        end if;
                end if;
         end if;
end process cnt_p;
   sink_sop <= '1' when cnt = "00000000000" else                '0';
   sink_eop <= '1' when cnt = "01111111111" else                '0';
end Behavioral;
```

Figura 4.23 Programación del modulo de control: F2: cfft

Una vez realizado el cálculo de la FFT, comienza el proceso de adecuación de los resultados para su posterior representación en un monitor externo. Para este fin se resuelve diseñar dos módulos en serie: A6*: General_mod* y A7:*General_equiv*.

Para la compresión de las operaciones matemáticas ejecutadas en ellos; se hace indispensable introducir como se realiza el dibujo del espectro en la pantalla (Figura 4.24).

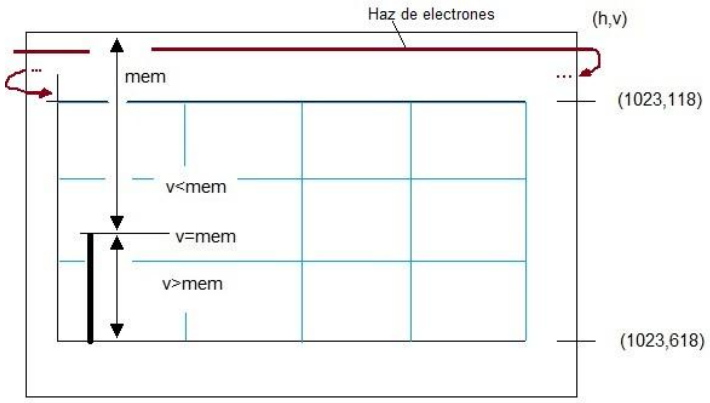


Figura 4.24 Representación esquema de dibujo del espectro

Como ha sido comentado en secciones anteriores el haz de electrones del monitor recorre la pantalla realizando una barrido de cada fila; cuando llega al último pixel salta a la siguiente fila y prosigue su recorrido hasta completar el total de píxeles.

En el diseño realizado los resultados que salen de la FFT son procesados para conseguir un valor equivalente al número de pixeles contenidos entre el borde superior de la pantalla (v=0) y el extremo superior de la barra de espectro (v=*mem*). De esta forma conforme el controlador SVGA va recorriendo la pantalla, si el índice de fila es mayor que *mem* al valor del pixel en cuestión se le asigna un nivel lógico alto, de lo contrario se mantiene en bajo.

Las operaciones matemáticas aplicadas a las componentes espectrales para conseguir *mem* se detallan en los módulos siguientes.

4.3.5.6 Módulo

A la salida de la FFT se obtienen los resultados de la parte real e imaginaria. Durante el proceso de cálculo por el algoritmo Radix-2 el número de bits de cada dato aumenta en (log_2N+1), siendo N=número de puntos de la FFT. Por ello se debe extraer los 12 bits que contienen la información necesaria de las componentes espectrales. Seguidamente, dado que el interés se centra en el cálculo de la magnitud, el primer paso es calcular el módulo. La operación realizada es $(x^2+y^2)^{1/2}$, siendo x la componente real e y la componente imaginaria.

Estructuralmente la programación englobada en el bloque A6: *General_mod*, se realiza en dos módulos en serie: M1: *scuad*, donde se realiza una multiplicación de cada dato por sí mismo; se eleva al cuadrado, y M2: *mRaiz*, donde se calcula la raíz cuadrada. Para la

implantación de este último módulo se ha utilizado un *MegaCore* configurado conforme la precisión de palabra requerida para minimizar la pérdida de información.

4.3.5.7 Equivalencia

El analizador de espectros dispone de una doble opción de visualización de la magnitud del espectro: lineal y logarítmica. Ambas están controladas por la señal *V_db*, de tal forma que si *V_dB='0'* se elige la escala logarítmica; si *V_dB='1'*, la lineal.

El diseño realizado consta de los siguientes dos módulos:

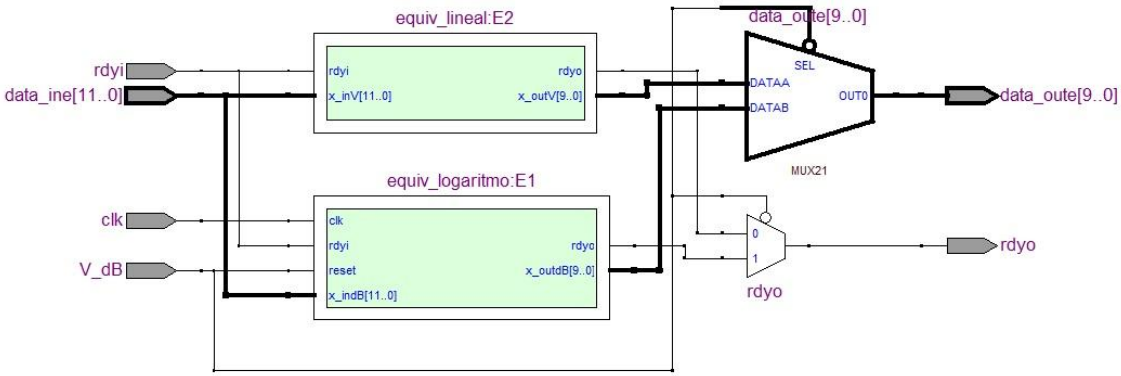


Figura 4.25. Estructura módulo A7:General_equiv

El control se realiza de dos formas: Por un lado la señal *V_db* actúa como reset del módulo E1: *equiv_logaritmo* y por otro como selector de un multiplexor cuyas entradas son las respectivas salidas de cada módulo. La señal *rdy* sirve para controlar la memoria RAM posteriormente, habilitando la escritura cuando los datos están presentes en la salida.

A continuación se detallan los cálculos ejecutados en cada uno de los módulos:

E1: *Equiv_lineal*

Los datos obtenidos de la FFT se escriben en una precisión de 12 bits en una codificación de punto fijo, cuyo valor máximo ("*011111111111*")*b* en decimal es $(2^{11}-1)=2047$. Dado que el valor máximo en decimal a obtener una vez calculado el módulo es de 2.5 (para una señales de entrada de amplitud 5V); a través de una simple regla de tres se obtiene que el dato en decimal toma el valor: *D*(2.5/1024)*, siendo *D* la salida del módulo de la FFT. Analizando la Figura 4.24 se ve que el número de filas limitadas para el dibujo es de 500 y dado que se tienen 2.5 valores a representar, el número de píxeles que se requiere para dibujar una barra espectral de magnitud unitaria es: *M=550/2*. Con estos dos datos se completa la siguiente ecuación tal que:

Dado *x_inV* la entrada y *x_outV=mem* la salida: $x\_outV = 618 - (\frac{2.5}{1024}) * (\frac{500}{2.5}) * x\_inV$

E2: *Equiv_logaritmo*:

En el primer paso se le aplica el logaritmo en base 10 al dato de entrada. Partiendo de la justificación del punto anterior respecto el valor del dato de salida del módulo la FFT se tiene:

$$\log_{10}(x\_indB) = \log_{10}(\frac{2.5}{1024} * X) = \log_{10}(2.5) + \log_{10}(X) - \log_{10}(1024)$$

Donde *X* representa el valor decimal en amplitud del espectro para una señal de entrada máxima de 5 V; para la cual, *Xmax*=2.5. Despejando se obtiene:

$$\log_{10}(X) = \log_{10}(x\_indB) + \log_{10}(2.5) - \log_{10}(1024)$$

Para calcular el dato *mem* que posteriormente se escribe en la memoria RAM y sirve para el dibujo del espectro se hacen las siguiente operaciones:

$$mem = 218 - 8 * 20 * \log_{10}(X)$$

Donde el factor 20 representa el factor de conversión en dB y el factor 8 equivale al número pixel que hay en una unidad de la escala en dB (se representa una banda de magnitud (-50,10)dB en 480 píxeles; (*480/60=8)*. La constante 218 representa la fila que marca 0dB en la pantalla de dibujo.

4.3.5.8 Memoria RAM

Por último se usa una memoria RAM capaz de almacenar 512 muestras dado que el espectro en frecuencia es simétrico y se va a representar únicamente la parte positiva. El tamaño de palabra es de 10 bits y la escritura se habilita mediante la señal *full*, que proviene del módulo anterior (*rdy*), realizando la labor de bandera. Cuando se detecta un flanco de subida del nivel lógico las muestras se escriben en el orden que entran; cuando el flanco es de bajada se deja de escribir. Para la lectura una señal contador proveniente del módulo de dibujo habilita el dato de la posición señalada.

4.3.5.9 Módulo de enable

La última parte de diseño del bloque analizador corresponde a la señales de habilitación del sistema. La programación propuesta resuelve por un lado la diferencia de tiempos de procesamiento de cada uno de los módulos y por otro la forma de generar el primer impulso para habilitar el sistema por primera vez.

Para ello el módulo dispone de dos entradas de 1 bit: *DataRdy* y *d_fft* y una salida *d_fifo* que se conecta al *reset* del A1: *conv_bi* y a la señal de inicio de escritura de la FIFO. *DataRdy* proviene del controlador del conversor y marca la entrada de cada dato. En el módulo se programan dos procesos:

1. En el primero se crea un contador de tal forma que cuando se reciba el primer pulso se active una señal auxiliar *aux* que se desactiva al siguiente pulso

2. El segundo está controlado por la señal auxiliar *aux* y la señal de entrada *d_fft* proveniente de la FFT *source_eop,* que marca el fin de salida del paquete de resultados. Ambas se conectan a la entrada de un multiplexor junto a un '1' lógico de tal forma que si *aux='1'=> d_fifo=aux*. Este caso marca el pulso de inicio sistema. Cuando *aux='0'=> d_fifo=d_fft;* es decir una vez el proceso de cálculo de la FFT ha terminado se da el inicio de entrada de los nuevos 1024 datos.

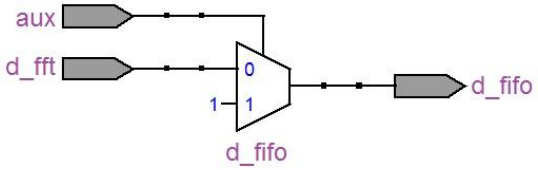


Figura 4.26. Estructura lógica del módulo A9*: Enable*

4.3.6 Dibujo SVGA

Es el módulo encargado de generar las señales que permitan la visualización gráfica en un monitor externo del análisis espectral en frecuencia, siguiendo la norma SVGA que divide la pantalla del monitor en una cuadrícula de 1024x786 píxeles. Para tal objetivo, los resultados obtenidos por el algoritmo de la FFT han sido adecuados en módulos anteriores.

En el diseño realizado, se hace una división entre dibujo estático y dinámico. La parte estática corresponde a ejes, cuadrícula y datos numéricos de amplitud y frecuencia (texto), implantados en el módulo D4: dib_texto y en D3: dib_espectro_ejes. En contraposición los datos dinámicos corresponden a la amplitud de las barras de espectro, que varían en el cálculo realizado por la FFT. El módulo encargado de su procesado se encuentra dentro de D3: dib_espectro_ejes.

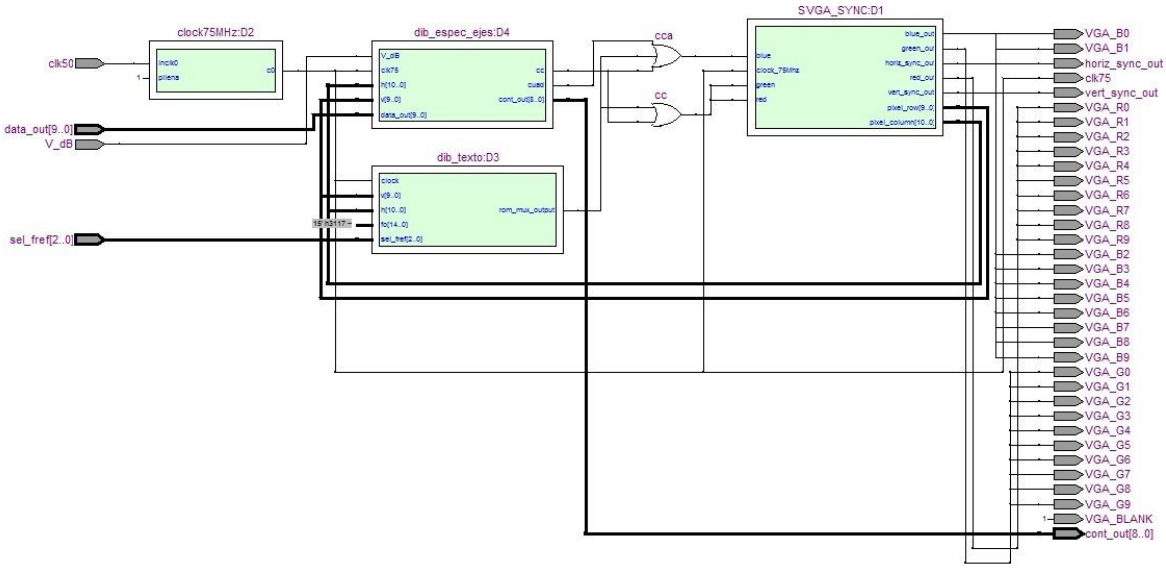


Figura 4.27 Estructura del módulo P3:Dibujo_SVGA

Como se aprecia en la Figura 4.27, se implantan otros dos módulos: D2: clock75MHz, es un reloj PLL que trabaja con un reloj base de 50MHz y tiene como salida 75MHz; y D1: SVGA_SYNC, se encarga de generar las señales de la norma de video SVGA.

Antes de dar paso al despiece de las diferentes partes del módulo; se hace necesario introducir la forma de dibujar en un píxel. Para ello se necesita de una señal de 1 bit, que indica la activación (valor lógico '1') o no de un color. A lo largo del proceso se generan varias señales que provienen de las capas correspondientes a la generación de los ejes de coordenadas, de los caracteres a escribir en pantalla, de la cuadrícula de color y de la magnitud de las barras del espectro. Estas señales mediante lógica combinacional se conectan a las entradas de color red, green y blue del módulo SVGA_SYNC, que controlará la conformación de la imagen en pantalla.

A continuación se desglosa cada uno de los módulos que generan las señales de dibujo para un análisis en mayor profundidad.

4.3.6.1 Dibujo espectro y ejes

En este módulo se generan las señales para el dibujo de las barras de espectro y de los ejes y cuadrícula mediante tres módulos tal que: uno genera las barras y los otros dos los ejes y cuadriculas en función de la opción elegida por el usuario: lineal o logarítmica.

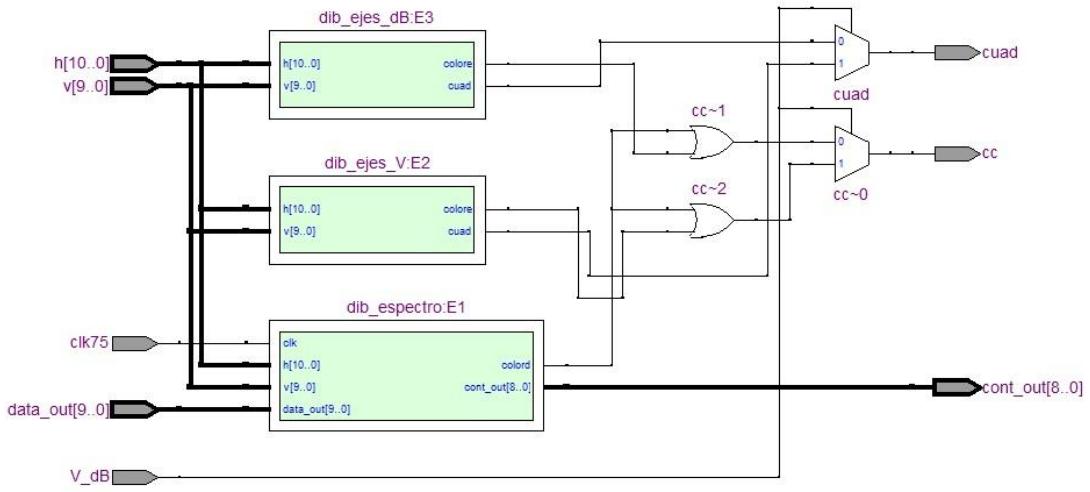


Figura 4.28 Estructura módulo D4: dib_espec_ejes

4.3.6.1.1 Dibujo espectro

Las dos señales principales del proceso son *data_out* y *cont_in*; que establecen una conexión con la memoria RAM. Por otro lado para realizar el control se tiene *clk* y las dos señales que marcan la dirección exacta (columna y fila)-(*h* y *v*) del píxel por donde se está haciendo el barrido SVGA. Mediante una sentencia *if* se limita el área de actuación a una cuadrícula de (0-997)x(0-618) pixeles (véase Figura 4.24).

El proceso, sincronizado con el flanco de subida del reloj *clk*, inicializa el valor de la señal *cont* (posición en la memoria RAM del dato requerido) y de la variable *aux* (columna en la pantalla en que se dibuja la barra cuya amplitud señala *data_out*) a cada barrido horizontal efectuado.

Durante el barrido y dentro de la cuadrícula limitada; partiendo de la columna número 17 se compara el valor de *data_out* con el de la fila en la que se encuentra. Si la columna es mayor que el dato se debe dibujar, sino se deja en negro (véase Figura 4.24).

Adicionalmente se aumenta la posición del dato de la memoria siguiente y el de la columna siguiente en dos; con el fin de dibujar las barras cada dos píxeles.

La señal *colord* es la utilizada para indicar la activación de un color a dibujar. La programación del cuerpo de arquitectura diseñada de este último se muestra en la siguiente Figura 4.29.

```
architecture Behavioral of dib_espectro is
 signal  c          :std_logic;
 signal  cont      : std_logic_vector (8 downto 0):=(others => '0');
begin

process (v,h,clk,data_out, cont )
  variable aux:  std_logic_vector(10 downto 0):=(others => '0');  --20
begin
   if clk='1' and clk'event then
          if h=0 then
                    aux:="00000010001";     --17
                    cont<="000000000";
          end if;
          if h=aux and v<618 and cont<="111101010" then          --490
                    aux:=aux+"00000000010";
                    if v>data_out then
                              c<='1';
                    else
                              c<='0';
                    end if;
                    cont<=cont+'1';
          else
                    c<='0';
          end if;
   end if;
end process;
          cont_out<=cont;
          colord<=c;
end Behavioral;
```

Figura 4.29 Programación del cuerpo de arquitectura del módulo E1: dib_espectro

4.3.6.1.2 Dibujo ejes y cuadrícula según escala lineal y logarítmica y conexiones

Los dos módulos E2: *dib_ejes_dB* y E3: *dib_ejes_V* (Figura 4.28) se encargan del dibujo de los ejes y cuadrículas mediante sentencias *if* en función de las condiciones impuestas a las señales *h* y *v*. Para la asignación de diferentes colores al dibujo se utiliza dos señales diferentes: *cuad* (color azul para la cuadrícula) y *colore* (color blanco para los ejes).

Se hace hincapié en la forma de conexión propuesta para las salidas que indican la activación de un color o no. Para evitar solapamientos en el dibujo de las barras del espectro y los ejes y tener una única señal (*cc*); las señales de salida de los ejes (*colore*) de cada módulo se conectan junto a la de las barras (*colord*) a una puerta lógica OR. De esta manera, si una de las dos señales esta a nivel lógico alto, la salida estará también en alto. Las señales para el dibujo de la cuadrícula a color se mantienen. Por último para la elección de escala se utiliza un multiplexor cuya señal de control es *V_dB.*

De esta forma se tienen dos señales a la salida del módulo D4: *dib_espectro_ejes*: *cc* y *cuad.* La señal *cc* representa el dibujo de los ejes y de las barras de espectro; mientras que *cuad* representa el dibujo de la cuadrícula de color.

4.3.6.2 Dibujo texto

Este módulo se encarga de generar la señal *rom_mux_output* que permite el dibujo del texto. Para ello se diseña una memoria ROM de un solo puerto de dimensiones: 256 palabras y 8 bits por palabra, mediante un *MegaCore* parametrizable. La forma de almacenar los caracteres que serán empleados es en matrices de 8x16.

El proceso se ejecuta en dos módulos interconectados que se describen a continuación:

4.3.6.2.1 T1: *General_rom_num*

Los caracteres almacenados en la memoria RAM se encuentran distribuidos en matrices de 16 filas. La señal *rom_address* representa el índice de cada fila; por tanto cada 16 filas se tiene un carácter diferente. En base a esto, se define la señal *character_address* como el índice que numera cada uno de los 16 caracteres que contiene la ROM.

Para diseñar un carácter se visualiza la matriz de 8x16 como una cuadrícula donde cada bit representa un recuadro. Asignar a un bit un 1 es asignar a un pixel color; por el contrario un 0 lo mantiene oscuro.

Por último se definen dos nuevas señales:
- *font_row:* representa dentro de la matriz de 8x16 el índice de cada fila
- *font_col:* representa dentro de la matriz de 8x16 el índice de cada columna.

Ya se tienen todas las señales de gobierno definidas. Ahora se explica el funcionamiento del algoritmo. Supóngase que se quiere representar un carácter. Para ello, se asigna un valor a la señal *character_address* que coincida con el índice de la matriz donde se encuentra definido. A través de sentencias condicionales del estilo *if*, se limita el área de la

pantalla donde se desea dibujar mediante las señales de barrido *h* y *v*. De este modo *font_row* toma el valor de los 4 LSB de la señal *v* y *font_col* los 3 LSB de la señal *h*. Para identificar el índice de la fila de la memoria ROM la señal *rom_address* corresponde a la combinación de *character_address&font_row*. La señal que representa la presencia de color toma el valor guardado en cada posición de la ROM.

Para entender mejor lo descrito se anexa el siguiente ejemplo donde la matriz que define el carácter tiene dimensión 8x8.

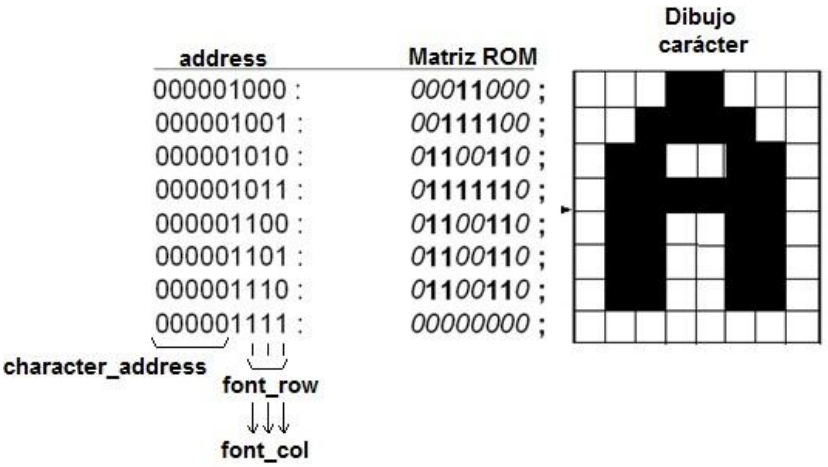


Figura 4.30 Ejemplo de las variables utilizadas para el dibujo de caracteres desde una memoria

Un último detalle es que se aprovecha una propiedad de la codificación en punto fijo; debido a que se trabaja con potencias de 2, despreciando los 4 LSB de la señal *address* se tiene definida la señal *character_address*, ya que:

$16=2^4$	001 0000
$32=2^5$	010 0000
$48=2^4+2^5$	011 0000
$64=2^6$	100 0000

4.3.6.2.2 num_txt_frec

Es el módulo encargado de asignar el valor a la señal de 1 bit *rom_mux_output*, que asigna el color de los pixeles a través de las señales que indican el carácter a dibujar y el espacio en la pantalla. Utiliza las siguientes dos asignaciones:

```
rom_address      <= character_address & font_row;
rom_mux_output   <= rom_data ( (conv_integer(not font_col(2 downto 0))));
```

En la pantalla se dibujan los textos correspondientes a las referencias de frecuencia de la cuadrícula en azul y el texto de la opción de enventanado en blanco.

4.3.6.3 SVGA_SYNC

Es el encargado de generar la norma de video SVGA propia del monitor de tal modo que genera las señales de control que se conectan directamente al FPGA y al conector VGA. Las señales de salida de este módulo son:

- *Horiz/vert_sinc_out*: señales de sincronismo horizontal y vertical. Indican el cambio de fila y de pantalla respectivamente.
- *VGA_BLANK*: Si esta a 0 hace que no saque nada por pantalla; si vale uno funciona normalmente.
- *Clk75*: Reloj lógico a la frecuencia de 75MHz
- *VGA_R[0:9], VGA_G[0:9], VGA_B[0:9]:* Nivel de rojo/verde/azul del pixel (0 a 1024). En este diseño se cortocircuitan limitando el número de colores a 8.

Adicionalmente el módulo proporciona, a través de sus salidas *píxel_column y píxel_row*, la dirección exacta (columna y fila) del píxel por donde se está haciendo el barrido SVGA; es decir, el píxel que está siendo visualizado, lo cual es necesario en el funcionamiento de otros módulos inmersos en el procesamiento de dibujo.

El módulo consta de 4 entradas: una para el reloj de frecuencia 75MHz y tres para los colores: green, blue y red. En el diseño realizado se hace uso de dos colores: el blanco y el azul. Atendiendo a ello y conforme la paleta de colores que marca la norma (Figura 2.22) se tiene:

Señal	Red	Green	Blue
Blanco	1	1	1
Azul	0	0	1
Negro(sin color)	0	0	0

Para la conexión de las salidas del resto de módulos a la entrada de este se utiliza dos puertas lógicas OR.

1. La primera tiene de entradas las señales de color blanco provenientes del dibujo del texto y de los ejes y barras de espectro. Su salida se conecta a las señales *green* y *red.*
2. La segunda tiene de entradas las mismas que la anterior junto a la que representa la cuadrícula y cuyo color es azul. Su salida se conecta a la señal blue.

Con esta conexión se consigue dar preferencia al dibujo en blanco. Cuando alguna señal que represente este color este a nivel alto red, green y blue estarán en alto y el monitor mostrara un punto blanco. En cambio el dibujo en azul únicamente se representa cuando corresponde al dibujo de la cuadrícula (*cuad*) y no se superpone ninguna señal blanca.

Capítulo 5

Pruebas y resultados

Para validar el funcionamiento del analizador de espectro sobre FPGA diseñado e implantado; en primer lugar se han realizado pruebas a cada una de las partes de las que consta la estructura. De esta manera, una vez comprobados los resultados teóricos con los obtenidos, se da paso al montaje del dispositivo final y al proceso de pruebas.

Las pruebas y simulaciones de cada una de las partes se describen a continuación.

5.1 Simulación de la placa de acondicionamiento de la señal y conversor A/D

Una vez realizado el diseño teórico de la etapa de acondicionamiento de la señal, se monta físicamente en una placa de pruebas (Figura 5.1).

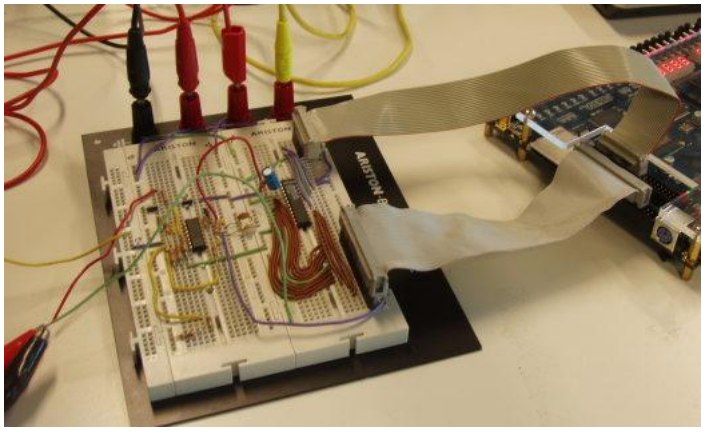


Figura 5.1 Instantánea etapa acondicionamiento de la señal

La alimentación se soluciona con dos fuentes de tensión que suministran ±5V y ±15V. Se usa un generador de señales que permite elegir entre distintos tipos de señal; así como variar su amplitud, frecuencia e introducir un offset. Las tierras de los tres dispositivos se cortocircuitan y se conectan con la placa.

Para visualizar los resultados se hace uso de un osciloscopio digital. Una vez comprobado el correcto funcionamiento se incorpora el conversor A/D y se conecta con el resto del circuito.

Para simular su respuesta y comprobar que es correcto se recurre a dos vías:

- Se conecta al FPGA. Mediante un sencillo algoritmo se crean las señales de control CLKIN y RD. A continuación se introduce una señal a través del generador de señales y con el osciloscopio se visualiza la forma de onda del pin BUSY del conversor.

- Se implementa el controlador del conversor programado para el proyecto y la salida se conecta a los leds disponibles en el kit DE2; de tal modo que cada uno de los 12 bits de salida del conversor son un led. Se programa el software en el FPGA y se pone en funcionamiento. Como señal de entrada se introduce una componente de directa regulable y se comprueba el valor de tensión aplicada con el número en binario representado por los leds.

  Con este procedimiento se evalúa el correcto funcionamiento del conversor y del controlador programado y se precisa la resolución.

  Terminada esta fase, se diseña el circuito mediante el software DesignSpark para crear una placa impresa.

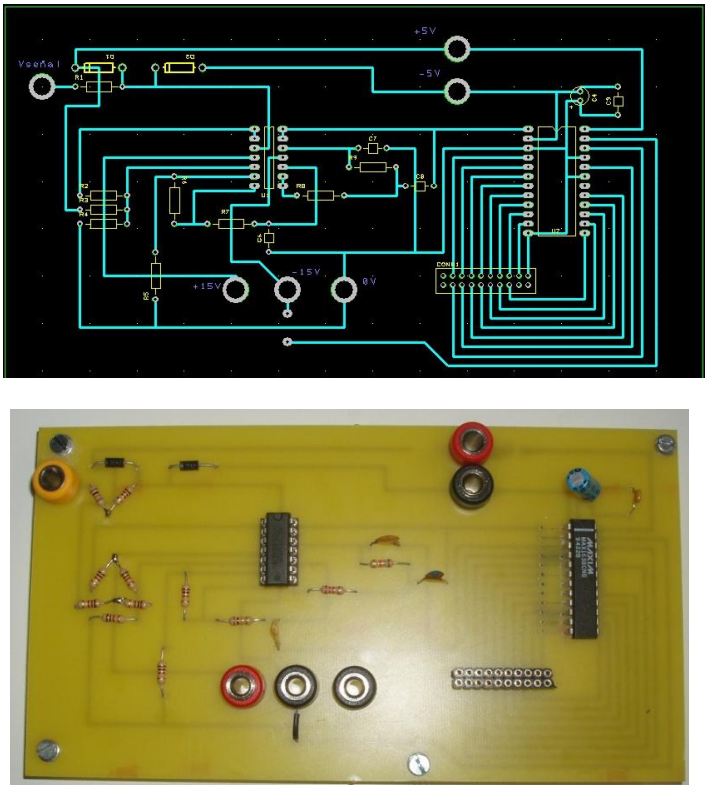


Figura 5.2 Esquema e instantánea de la PCB del circuito electrónico de la etapa de acondicionamiento y conversión A/D

5.2 Simulación del módulo P2: *Analizador* programado en Quatus II

En la realización de las simulaciones se trabaja con Quartus, ModelSim y con Matlab. De esta forma se obtienen unas evaluaciones precisas de los resultados.

Mediante el uso de la librería [`LIBRARY std; USE std.textio.all;`] es posible introducir valores en la simulación de la programación VHDL utilizando Quartus II y ModelSim. De esta forma, en Matlab se muestrea una señal y se convierte a binario. Los datos se guardan en formato .txt, que puede leer Quartus II. Del mismo modo el proceso inverso es factible para simular los resultados en Matlab (Figura 5.4) una vez han sido procesados por el sistema programado (ver Anexo II).

La siguiente Figura 5.3 muestra una simulación realizada a través de ModelSim. Esta herramienta ha sido de gran utilidad en todo el proceso de diseño y programación, posibilitando resolver los problemas y comprobar el comportamiento de las señales, variables y contadores.

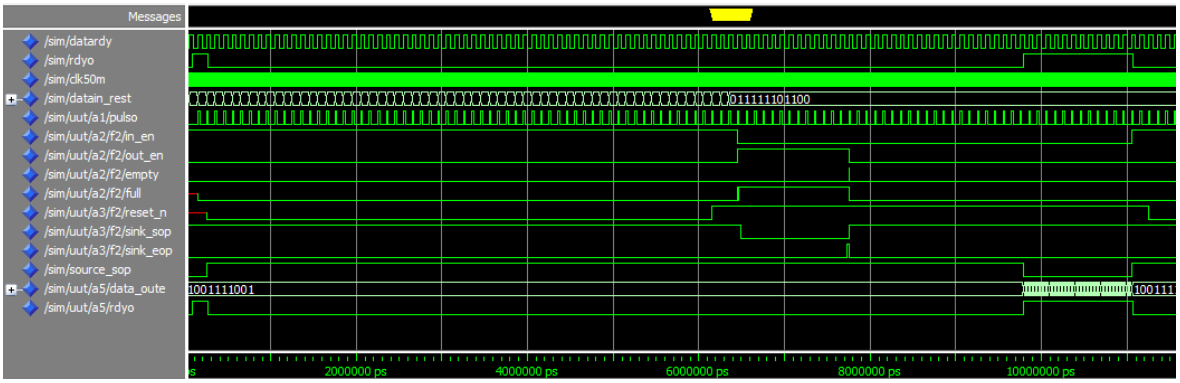


Figura 5.3 Simulación en ModelSim del módulo P2:Analizador

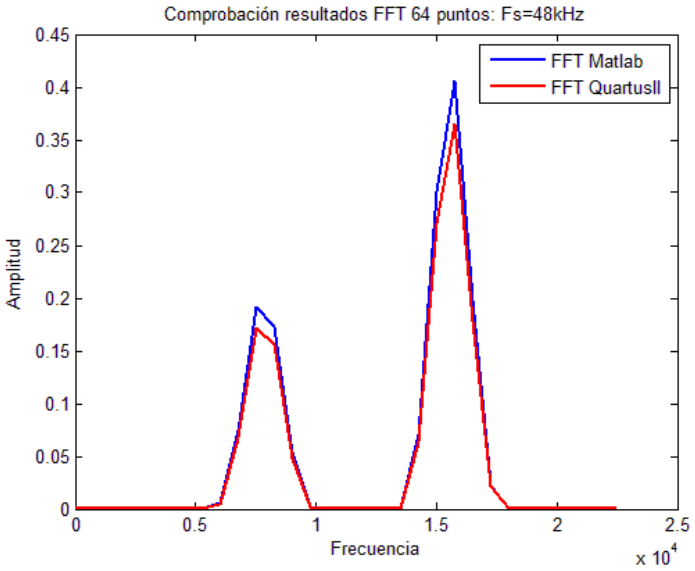


Figura 5.4 Gráfica obtenida en Matlab, donde se comparan los resultados teóricos con los obtenidos en el simulador ModelSim de QuartusII.

5.3 Simulación del módulo P3: *Dibujo_SVGA*

Para la comprobación del módulo de programación que dibuja el espectro en un monitor externo, se crea una memoria ROM que almacene 500 valores fijo simulando el cálculo realizado por la FFT, de este modo las muestras pasan por los módulos de equivalencia y dibujo y se representan en el monitor para comprobar su funcionamiento. El mismo proceso se sigue a la hora de diseñar los dibujos del texto, cuadrícula y ejes.

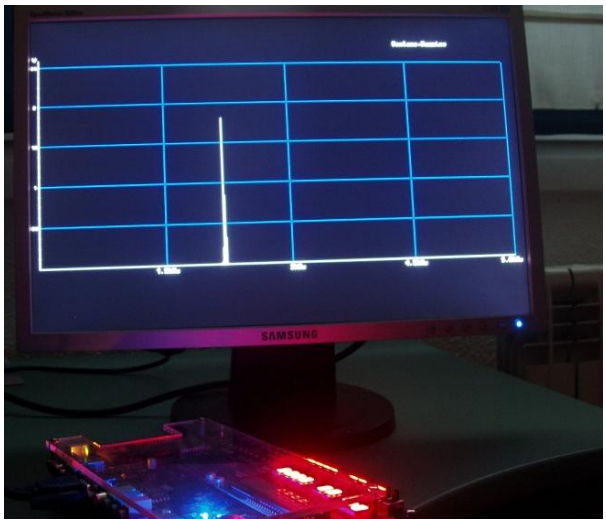


Figura 5.5 Instantánea de la comprobación del módulo P3: Dibujo_SVGA

5.4 Recursos ocupados en el FPGA

Una vez finalizada y comprobada la programación en VHDL se compila el diseño general en Quartus para su implementación en hardware, obteniendo los siguientes resultados acerca de los recursos ocupados en el FPGA.

Recursos	Relación Uso/Disponibilidad	Porcentaje de uso
Total elementos lógicos	21753/33216	65%
Total funciones combinacionales	19477/33216	59%
Registros lógicos dedicados	11547/33216	35%
Memoria bits	79404/483840	16%
Multiplicadores embebidos de 9 bits	58/70	83%
Total PLLs	1/4	25%

Figura 5.6 Tabla recursos totales ocupados en el FPGA

En vista a los resultados de la Figura 5.6 sería posible la implementación de un algoritmo de la FFT de más puntos para ampliar el potencial de este diseño logrando mejores resultados. Análogamente se identifica el bloque que más recursos supone como el banco de filtros digitales (Figura 5.7)

Recursos	Relación Uso/Disponibilidad	Porcentaje de uso
Total elementos lógicos	12293/33216	37%
Total funciones combinacionales	11063/33216	33%
Registros lógicos dedicados	7524/33216	22%

Figura 5.7 Tabla recursos ocupados por los filtros FIR en el FPGA

Atendiendo a la Figura 5.7 se puede llegar a un compromiso entre precisión en el filtrado y recursos ocupados; permitiendo el uso de dispositivos FPGAs mucho menos potentes.

5.5 Resultados obtenidos

Una vez comprobadas todas las partes de las que consta el proyecto se lleva a cabo el montaje final. Mediante el generador de señales se configuran diferentes tipos de pruebas para probar el sistema. Asimismo se analiza con cuidado la posible existencia de aliasing; llegando a la conclusión de que no se presenta en ningún caso. Para comprobar se usa el software Matlab, un osciloscopio digital que nos muestra el espectro y señales de alto contenido armónico como señales cuadradas, triangulares, pulsos y dientes de sierra.

En la Figura 5.8 y 5.9 se puede observar el montaje final del analizador de espectro

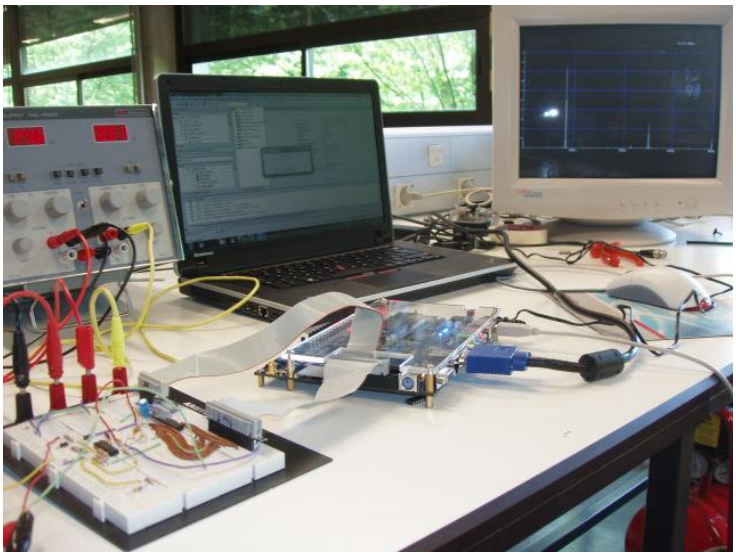


Figura 5.8 Instantánea del montaje final

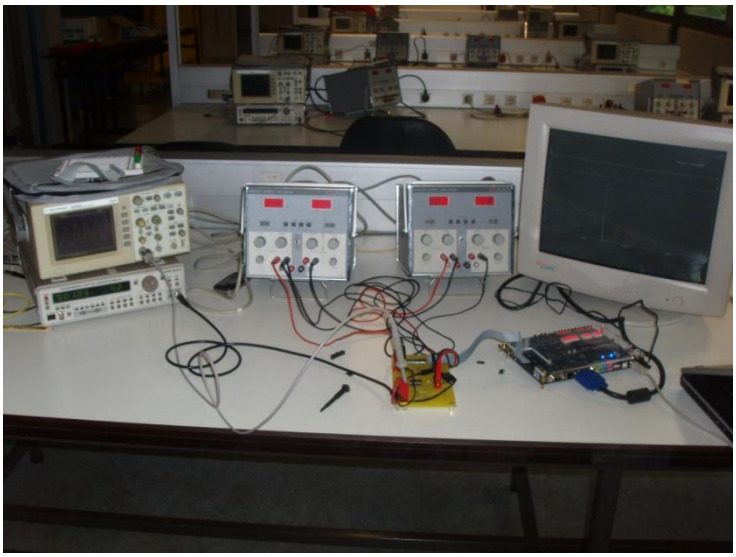


Figura 5.9 Instantánea del montaje final

Capítulo 6

Análisis económico

En todo proyecto o investigación la valoración económica es esencial. Permite hacer la comparación entre gastos y beneficios para determinar la estrategia que conducirá a la optimización y la eficiencia del proceso de investigación y obtención de resultados. En este trabajo se valoran los gastos por concepto de material y consumo eléctrico, los cuales son reflejados a continuación.

Material	Precio
Altera® DE2 BOARD	213 euros
MAX163	16 euros
Monito PC	13 euros
Componentes eléctricos*	3 euros
Placa pruebas cableado	10 euros
TOTAL GASTOS DE MATERIAL	255 euros

*Diodos,resistencias,condernamplificador operacional

Material	Tiempo [h]	Potencia [W]	Consumo[kWh]
Odernador Personal	1500	65	97.50
Lámpara	250	20	5
Altera® DE2	20	1	0.001
Generador señales	2	80	0.16
Fuentes de alimentación	2	80	0.16
TOTAL CONSUMO ELÉCTRICO			102.82 kWh
TOTAL GASTO ELÉCTRICO [0,142138euros/Kwh]			14.70 euros

IVA gasto material (18%)	45.90 euros
IVA consumo eléctrico (18%)	2.65 euros
TOTAL:	**318.25euros**

Conclusiones análisis económico

El gasto incurrido en la realización del proyecto asciende a 320 euros; si se compara con un analizador digital de espectro comercial en el mismo rango de frecuencias cuyo precio parte de 500 euros, se cumple el objetivo planteado de bajo coste.

Adicionalmente, se hace constar que la mayor suma de capital se debe al Kit de Desarrollo y Educación DE2 junto a la FPGA embebida. Partiendo de la base de que su uso se puede ampliar para otros muchos aspectos educativos en centros universitarios como son prácticas, proyectos, creación de nuevos IP Cores, etc; el coste asumido no es elevado.

En especial, supone una reducción enorme de capital si se introduce la posibilidad de representar una señal en tiempo real; es decir, incorporar la programación para el diseño de un osciloscopio. Disponiendo de esta forma de dos equipos en uno.

Por otro lado se dispone de un instrumento de medición electrónico complejo como lo es un analizador digital de espectros, que puede constituir la base para el desarrollo de nuevas investigaciones afines.

Horas de trabajo

	Cuba	España	Total
Horas de trabajo	400 horas	350 horas	750 horas

El total de horas de trabajo invertidas en la ejecución del proyecto fin de carrera asciende a 750 horas de trabajo. Dentro de estas horas se engloba:

- 200 horas : tiempo dedicado a la búsqueda y estudio de electrónica digital, analizadores de espectro y demás aspectos incluidos en el proyecto.
- 400 horas : programación realizada en VHDL y simulación de los resultados.
- 50 horas : diseño e implantación etapa de acondicionamiento de la señal
- 20 horas : realización de pruebas
- 80 horas : redacción memoria

Conclusiones

En este proyecto fin de carrera se presentó la creación de un analizador de espectro a bajo coste en el rango de frecuencia de 0kHz a 20kHz mediante la programación de hardware digital implantado en un FPGA. De este modo, una vez terminado y realizadas las pruebas oportunas se dan por validos los objetivos iniciales propuestos.

- Durante toda la ejecución del proyecto se ha hecho una revisión de los fundamentos teóricos relacionados. Buscando diferentes vías para la resolución de los problemas inherentes al diseño y realizando pruebas y simulaciones.
- Se ha completado el diseño y la implantación de la etapa de acondicionamiento, minimizando el uso de componentes. Finalmente se ha realizado un diseño PCB con el fin de disminuir el ruido en la señal y ampliar los conocimientos.
- Se ha desarrollado e implementado la programación en VHDL de una manera exitosa; así como de los algoritmos para el control de video un monitor externo mediante la norma SVGA.
- Se han explorado diferentes opciones habilitadas en los kit de desarrollo donde se encuentra el FPGA; haciendo simulaciones con los leds, botones, interruptores y memorias.
- Se han implantado en el sistema prestaciones adicionales como el zoom espectral y la elección de enventanado.
- Se ha diseñado un dispositivo de análisis espectral que cumple los objetivos de bajo coste respecto a sistemas comerciales.
- Se ha ganado experiencia en el uso de los software: Quartus II, ModelSim, Xilinx ISE, Matlab, Proteus, DesignSpark y FilterPro.

Adicionalmente el haber dispuesto de dos FPGAs embebidas en dos kit de empresas distintas y a la vez punteras en el diseño y fabricación de estos dispositivos digitales, como son Altera y Xilinx; ha posibilitado el conocimiento de las diferentes características que ofrecen cada uno. Destacando especialmente las opciones disponibles en las herramientas de software (Quartus II y Xilinx ISE), los diferentes *IP-Cores* y la bibliografía y opciones da ayuda que ofrecen.

Recomendaciones y Trabajos futuros

Durante el tiempo de trabajo en este proyecto, mediante la experimentación y la documentación se han encontrado diversas posibilidades para la mejora en la precisión del instrumento de medida. De este modo se destaca:

1. En primer lugar el uso de un hardware más potente posibilitaría el trabajo con una FFT de más puntos; aumentando considerablemente la calidad del espectro. Asimismo el uso de un conversor bipolar con mayor resolución permitiría aumentar la precisión, incluyendo el trabajo con una codificación en punto flotante más acorde con el diseño realizado del *MegaCore* de la FFT y simplificando la etapa de acondicionamiento de la señal.

2. Desde el punto de vista de la programación se invitaría a una mayor depuración del código para minimizar el uso de recursos en el FPGA; así como habilitar una configuración parametrizable más sencilla de tal forma que mediante las diferentes consolas del kit de desarrollo se pudiese elegir: el ancho de banda y el número de puntos de la FFT.

3. A lo largo del proyecto se ha estudiado la posibilidad de utilizar la FFT para determinar la frecuencia fundamental de la señal. En base a ello, se establecen dos objetivos:

   - La creación de un IP Core que realice la función de autocorrelación.

   - La creación de un módulo eficiente que calcule la frecuencia fundamental de una señal periódica a través de la FFT.

4. En base a mejorar el sistema de variación de span o zoom espectral realizado en el diseño del proyecto se lanza el siguiente punto de actuación. Dado que al realizar el filtrado y enventanado de la señal únicamente se ofrece la posibilidad de realizar un zoom desde 0 a la frecuencia de corte surge la necesidad de tener la opción también valida de elegir un span entre una frecuencia mínima distinta de cero y la frecuencia de corte, mediante la especificación por parte del usuario de la frecuencia central *fc* y del ancho de banda

   En este sentido, la adaptación de la programación no sería demasiada costosa y con el diseño de unos pocos módulos estaría satisfecha, explotando la teoría de desplazamiento en frecuencia mediante la convolución por un coseno a *fc*

5. Otro punto futuro de trabajo se centra en explotar la conexión del kit con un PC para un procesado posterior de los datos.

Bibliografía

[Señales y Sistemas; Electrónica digital; Procesamiento digital]

- **[SyS.1]** Proakis, John G.; Manolakis, Dimitris G.: *Tratamiento Digital de Señales*. Ed. Prentice Hall. 1998
- **[SyS.2]** Oppenheim, Alan V.; Willsky, Alan S.: Nawab, S. Hamid. *Señales y Sistemas.* Ed. Prentice Hall
- **[ED.3]** Delgado, A.E.; Mira, L.; Dormido, S.: *Teoría de Electrónica Digital*. Ed. Sanz y Torres
- **[ED.4]** Wakerly, John F.: *Digital Design, Fourth Edición*. Prentice Hall, 2006
- Ifeachor, E.; Jervis, B.: *Digital Signal Processing: a practical approach*. Ed. Addison WesleyLongman. Ltd., Harlow, England. 2000
- Dr. Alvarado Moya, Jose Pablo: *Procesamiento Digital de Señales.* Escuela Tecnología Electrónica, Tecnológico Costa Rica. 2011
- I.E.C Francisco Javier Torres Valle: *Síntesis y Descripción de Circuitos Digitales utilizando VHDL*. Universidad Autónoma de Guadalajara. 2001
- Tocci, Widmer: *Sistemas digitales, principios y aplicacione.* Ed. Prentice Hall. 2003
- Rincón Pasaye, José Juan.: *Procesamiento Digital de Señales*. Facultad de Ingeniería Eléctrica. Universidad Michoacana de San Nicolás de Hidalgo

[VHDL]

- **[VHDL.1]** IEC Franciso Javier Torres Valles. *Síntesis y Descripción de Circuitos Digirales.* Universidad Autónoma de Guadalajara. España. 2001
- Freire Rubio, M.A.: *Introducción al Lenguaje VHDL*. Universidad Politécnica de Madrid. 2008
- Y.Sri Chakrapani, M.Tech. *VHDL AND CAD Laboratory*.
- Pardo, F.; Boluda, J. A.*: VHDL: Lenguaje para síntesis y modelado de circuitos*. Tercera Edición para España. Ed. RA-MA. 2011
- Perry, Douglas L.: *VHDL: Programming by example*. Fourth Edition. Ed. McGraw-Hill. 2012
- González Ayala, C. A.: *Metodología de Diseño y modelado de circuitos en VHDL*
- Sanchez Elez, Marcos: *Introducción a la programación en VHDL*. Facultad Informática. Universidad Complutense de Madrid

[FFT]

- **[FFT.1]** Luis Gerardo de la Fraga. *La Transformada Discreta de Fourier y la Transformada Rápida de Fourier*. 30 de mayo de 2001.
- **[FFT.2]** Transformada Rápido de Fourier. *http://www.tecnun.es/asignaturas/tratamiento%20digital/tema6.pdf*
- **[FFT.3]** Altera. *FFT MegaCore Function User Guide (ug_fft)*

[Filtros analógicos y digitales]

- Bautista Cuéllar, Ricardo Valerio: *Matlab y el diseño de filtros digitales*. Revista Digital "Investigación y Educación".
- Rader, C. M.; Gold, B.: *Digital filter design in the frequency domain*.
- Álvarez Cedillo, J.A.; Lindig Bos, K.M.; Martínez Romero, G.. *Implementación de Filtros Digitales Tipo FIR en FPG*.
- **[FA.1]**Texas Instrument. *Active Filter Design Techniques*

[1][Tesis; Artículos]

- Soberanis Garfias, Alejandro. *Diseño y construcción de un analizador de espectros usando una plataforma basada en FPGA*. Instituto Politécnico Nacional. México
- J. Valeriano; C. Ojeda; J. Castillo; S. Quintana. *Diseño e implantación de un frecuencímetro utilizando el CPLD*. Centro de Instrumentos UNAM
- Castro García, P.L.. *Desarrollo de un módulo digital para el análisis espectral de señales de audio*. Universidad Superior de Cataluña
- Torres, V.; Valls, J.; Perez, Asunción; Sansaloni, T.*: Diseño de un analizador de espectro en FPGA*. Universidad Politécnica de Valencia.
- Yelpo, Víctor; Costa, Diego; Páez, Carlos Sosa. *Módulo de cálculo de la Transformada Rápida de Fourier para analizador de espectros en tiempo real en FPGA*. Universidad Nacional de San Luis, Argentina.

[SVGA]

- *Controlador de la pantalla VGA*. Dep. Tecnología Electrica. Universida Rey Juan Carlos
- **[SVGA.1]** Lopez Portilla Vigil, Bárbaro M.; Iglesias Martínez, M.E.: *Implementación de controladores para periféricos del kit de desarrollo Spartan-3E de Xilinx*.Universidad Hermanos Saiz Montes de Oca, Pinar del Rio. Cuba

[Determinación frecuencia]

- Alvarado Reyes, J.M.; Stern Forgach, C.E.: *Un complemento al teorema de Nyquist*. Universidad Nacional Autónoma de México

- Terán, J.C.*: Analizador Lógico de tiempos implementado en una FPGA utilizando el bus PCI como interfaz de comunicación*. Instituto Tecnológico de Chihuahua, México,
- Nicolae, Mihai; Rugina, Ioan; Vasile, Alexandru. *Optimizing Implementation of Autocorrelation Function.*
- Escalante Olarte, Jaime. *Frecuencímetro Digital con Autoescalamiento*
- Alzate, Ricardo; Castellanos, Germán. *Implementación de algoritmos de estimación del pitch en tiempo real sobre porcesadores DSP*. Universidad Nacional de Colombia

[FPGA; datasheet]

- Spartan-3E FPGA Family: Complete Data Sheet.
- **[FPGA.1]** López Vallejo, M.L.; Ayala Rodrigo, J.L: *FPGA: Nociones básicas e implementación.* Universidad Politécnica de Madrid. España. 2004
- **[FPGA.2]** Altera Corporation. *Embedded Programmable Logic Family*. Data Sheet. 2000
- Hennire, F.C.: *Iteratuve Arrays of Logical Circuits.* MIT Press. 1961
- XILINX: *The Programmable Gate Array Data Book.KSan José, CA. USA. 1992*
- Pagina Web de la compañía Xilinx: *http:// www.xilinx.com*
- Spartan™-3E Starter Kit Board*: http://www.xilinx.com/s3estarter*
- Página web compañía Altera*: http://www.altera.com*
- Data Sheet: MAX163, TL084: *http://www.alldatasheet.es*
- Universidad de Vigo: *http://www.dte.uvigo.es/logica_programable*
- Analizador de espectro*: http://www.aaronia.es/productos/analizadores-de-espectro*
- Foro VHDL: *http://www.openhdl.com/vhdl*
- Foro electrónica: *http://www.foroselectronica.es*
- Página web Matlab: *www.mathworks.es/products/signal*
- Página web semiconductores*: http://www.national.com*
- Data Sheet Pmod AD1 : *http://www.digilentinc.com*
- Documentación Altera MegaCore: *http://www.altera.com/literature/ug*

ANEXO

I. Programación de los procesos en VHDL para lectura y escritura de un .txt

```
constant FIR_INPUT_FILE_c  : string := "C:\Users\Desktop\SIM\data_input.txt";
constant FIR_OUTPUT_FILE_c : string := "C:\Users\Desktop\SIM\data_output.txt";

stim_proc: process
    file in_file     : text open read_mode is FIR_INPUT_FILE_c;
    variable data_in :  bit_vector(11 downto 0);
    variable indata  : line;

 begin
          wait until rising_edge(clk);
          while not ENDFILE(in_file) loop
                    readline(in_file,indata);
                    read (indata,data_in);
                              DataIn <= To_UX01(data_in);  --binario
          end loop;
          wait;
end process;

sink_model : process(clk48kHz) is
   file ro_file   : text open write_mode is FIR_OUTPUT_FILE_c;
   variable rdata : line;
   variable data_r : integer;

begin
          if rising_edge(clk) then
                    dataOut := to_integer(signed(data_out));  --binario a decimal
                    write(rdata, data_r);
                   writeline(ro_file, rdata);
          end if;
 end process sink_model;
```

II. Función de un proceso iterativo de cálculo de la frecuencia fundamental (Matlab)

```
function [f,it] = detfrec(fo)
s = 1024;       %Cantidad de puntos de la FFT
n = [0:s-1];
f=20000;        %Frecuencia máxima del analizador
f1=0;
it=0;
while abs(f1-f)/f>=0.01  %estimador del error
   f1=f;
   fs=f*2.56;          %frecuencia de muestreo cumpliendo el teorema de Nyquist
   x = 4*cos(2*pi*fo*n/fs) + 2*cos(2*pi*3*fo*n/fs + pi/4);
   A = fft(x,s)/s;
   XX = abs(A);
   mayor = 0;
   for i = 1:length(XX)
      if XX(i) > mayor;
         mayor = XX(i);
         pos = i;
      end
   end
   if pos==1
      pos1=1;
   else
      pos1=pos-1;
   end
   f=pos1*fs/(s-1);%frecuencia estimada
   it=it+1;
end
sprintf('La frecuencia fundamental estimada es:')
f
sprintf('El número de iteraciones necesarias son: ')
it
```